Article

HALF: A High-performance Automated Large-scale Land Cover Mapping Framework

Jiarui Zhang [1,2], Zhiyi Fu [3], Yilin Zhu [1,2], Bin Wang [1,2], Keran Sun [1,2], Feng Zhang [1,2,*]

[1] School of Earth Sciences, Zhejiang University, 866 Yuhangtang Rd, Hangzhou 310058, P.R. China
[2] Zhejiang Provincial Key Laboratory of Geographic Information Science, 866 Yuhangtang Rd, Hangzhou 310058, P.R. China
[3] Institute of Remote Sensing and Geographical Information Systems, School of Earth and Space Sciences, Peking University, Beijing, 100871, P.R. China
* Correspondence: zfcarnation@zju.edu.cn; Tel.: +86-571-8827-3287

Abstract: Large-scale land cover plays a crucial role in global resource monitoring and management, as well as research on sustainable development. However, the complexity of the mapping process, coupled with significant computational and data storage requirements, often leads to delays between data processing and product publication, creating challenges for dynamic monitoring of large-scale land cover. Therefore, improving the efficiency of each stage in large-scale land cover mapping and automating the mapping process is currently an urgent issue to be addressed. We propose a high-performance automated large-scale land cover mapping framework (HALF) that introduces high-performance computing technology to the field of land cover production. HALF optimizes key processes, such as automated sample point extraction, sample-remote sensing image matching, and large-scale classification result mosaicking and updating. We selected several 10°×10° regions globally and the research makes several significant contributions: (1) We design HALF for land cover mapping based on docker and CWL-Airflow, which solves the heterogeneity of models between complex processes in land cover mapping and simplifies the model deployment process. By introducing workflow organization, this method achieves a high degree of decoupling between the production models of each stage and the overall process, enhancing the scalability of the framework. (2) HALF proposes an automatic sample points method that generates a large number of samples by overlaying and analyzing multiple prior products, thus saving the cost of manual sample selection. Using high-performance computing technology improves the computational efficiency of sample-image matching and feature extraction phase, with 10 times faster than traditional matching methods.(3) HALF proposes a high-performance classification result mosaic method based on the idea of grid division. By quickly establishing the spatial relationship between the image and the product and performing parallel computing, the efficiency of the mosaicking in large areas is significantly improved. The average processing time for a single image is around 6.5 seconds.

Keywords: Land Cover; High-performance computing; Remote sensing; Workflow; Automation

1. Introduction

Global Land Cover (GLC) plays a critical role in global resource monitoring and sustainable development research, as it provides vital insights into ecological diversity, carbon cycling, and human-environment relationships [1–3]. In recent years, with the rapid development of remote sensing technology, land cover classification based on remote sensing images has become the mainstream method for modern land cover mapping.

Unlike satellite images which are usually obtained in near-real-time, GLC products are typically associated with substantial lag times between the processing of images and the release of data [4]. The shortest production cycle of GLC products is usually one year, and seasonal or monthly datasets are relatively rare [5]. In addition, the higher the product resolution, the longer the production cycle. Currently, there are many GLC products

 doi:10.20944/preprints202305.0134.v1

updated annually, such as the Moderate Resolution Imaging Spectroradiometer (MODIS) Land Cover Type product (MCD12Q1) with a resolution of 500 meters provided by National Aeronautics and Space Administration (NASA) from 2001 to present [6]; Climate Change Initiative (CCI) program of the European Space Agency (ESA) provides a 300-meter dataset (1992-2018), and the Copernicus Global Land Service (CGLS) [7] provides a 100-meter dataset for land cover; Chen et al. [8] produced the GlobeLand30 with a resolution of 30 meters from Landsat and China's HJ-1 satellite images for 2000 and 2010. The latest version was launched for updating in 2017 and officially released in 2020. GLC with yearly or even higher intervals limits the monitoring capability of land cover dynamic changes. Hence, improving the computational efficiency of various stages in land cover mapping under big data and automating the mapping process currently become the key issues that need to be addressed. Larger scale, higher resolution, and faster update frequency have exponentially increased the amount of data, posing new challenges for storage and computing power. At the same time, the land cover mapping process is complex, and each stage depends on different models. How to improve the efficiency of each link in land cover mapping, systematize and standardize the mapping process, and achieve automated mapping of large-scale land cover gradually become one of the key issues.

Sample collection is considered the most time-consuming and labor-intensive process in producing GLC [9]. Existing methods for extracting classification samples still rely primarily on field investigations and visual interpretation of imagery. This approach not only demands extensive manual labor, but also restricts the study area's scale significantly. To address this problem, some scholars have studied the automatic extraction of classification sample points by combining prior products and auxiliary data [10–14]. This method uses classification rules extracted from prior products to quickly obtain a large number of spatially distributed and high-quality samples, thereby improving the efficiency of sample extraction and increasing the update frequency of land cover products. Zhang and Liu [15] used a total of 27,858,258 training samples in the production of the GLC, which is several hundred times higher than the number of training samples traditionally selected manually, occupying a large amount of storage and computing space. ESRI [16] even used a super-large dataset of Sentinel-2 with a total of more than 5 billion samples in the production of 2020 LandCover with a resolution of 10 meters. In the process of matching sample points and remote sensing images, the extraction of spectral features and indices of remote sensing images will consume a large amount of resources. The introduction of other auxiliary data such as elevation and climate [17] further exacerbates the computational pressure on the matching step.

Remote sensing image mosaicking is the process of stitching multiple remote sensing images into a single geometric-alignment composite scene for a wide field of view. Large-scale image mosaicking is a computationally intensive process [18]. Creating a national or global level product by mosaicking tens of thousands of image scenes that cover tens of terabytes of data could take several days or even weeks to complete [19]. Using high-performance computing methods to process each task in parallel can improve the efficiency of image mosaicking. Researchers have studied parallel mosaicking methods, Zhang et al. [20] combined aerial digital photogrammetry principles with parallel computing techniques to propose a parallel mosaicking method for massive aerial digital images. Chen et al. [21] improved an algorithm used for video image mosaicking to make it suitable for remote sensing images, overcoming the problem of multi-images having to be divided into one-to-one mosaics. Ma et al. [22] improved parallel mosaicking efficiency by creating a dynamic task tree to divide the processing order of remote sensing images. Remote sensing data, intermediate data, and result data will undergo frequent I/O operations during the computation process, which can lead to I/O bottlenecks. Jing et al. [23] customized RDD operators in Spark to accelerate the parallel mosaicking process through in-memory computation. Ma et al. [19] introduced a memory-based file management system called Alluxio, which is combined with compute-intensive remote sensing image mosaicking to effectively alleviate the I/O bottleneck in the mosaicking process. The process of mosaicking

 doi:10.20944/preprints202305.0134.v1

classification results is similar to that of remote sensing image mosaicking, but mapping after all data within the study area is collected may decrease the efficiency of dataset updating. Therefore, when mosaicking the classification results, it is necessary to consider that the acquisition time may vary between different regions.

The earliest remote sensing models were limited by the performance of desktop softwares running on a single machine. With the development of high-performance computing (HPC), some HPC-based methods have been considered as effective solutions for solving computational challenges, such as MPI/OpenMP [24], Hadoop [25], Spark [26], and CUDA programming supported by GPUs [27]. Many HPC frameworks targeting remote sensing and geographic information computing scenarios have also been proposed, such as SpatialHadoop [28], Hadoop GIS [29], GeoFlink [30], etc. Although using these frameworks and migrating computing to the cloud has alleviated the pressure of local computing, the sharing of data and models still presents difficulties due to different users' data storage and organizational formats [31]. The models for various stages of GLC mapping use different programming languages and run in different environments, which leads to researchers spending a lot of time and effort on setting up and maintaining the environment. Nüst et al. [32] have encapsulated runtime environment dependencies and analysis components in containers, improving the reusability, accessibility, and transparency of programs. Wang et al. [33] have implemented the automatic encapsulation of spatial information processing operators in different environments based on containers. Atomic-level geographic processing services have limited capacity and and may not suffice to support large-scale and complex processing tasks[34,35]. Thus, it is necessary to construct complex workflows for efficient resource allocation to facilitate data and model orchestration of complex GLC mapping[36].

To address the aforementioned challenges, this paper designed an high-performance automated large-scale land cover mapping framework(HALF) under a distributed architecture. This framework introduces high-performance computing technology into various stages of mapping and develops parallel computing strategies for automated sample production, classification result mosaic and updating based on data characteristics, effectively improving the computing efficiency in large-scale land cover scenarios. Furthermore, the models in each stage of the mapping process are encapsulated based on container technology and organized by Airflow, resolving the heterogeneity of the operating environment and facilitating data reuse. The HALF framework is empirically tested in several 10°×10° regions worldwide, providing theoretical and technical support for automated GLC producing.

2. Materials and Methods

2.1. Experimental Data

2.1.1. Landsat8

Landsat 8 is the eighth satellite in the United States Landsat program, carrying the Operational Land Imager (OLI) and Thermal Infrared Sensor (TIRS). The remote sensing images obtained by OLI include nine bands, with spatial resolutions of 30 meters for all bands except the panchromatic band, which has a spatial resolution of 15 meters. The remote sensing images obtained by TIRS include only two separate thermal infrared bands, with a spatial resolution of 100 meters. The data used in this article is the Level-2 Science Products of Landsat Collection 2, which are products obtained after radiometric correction and system-level geometric correction. The data for 2015 was selected, and for each scene, the image with the smallest cloud cover within a year was chosen. Figure 1 shows the number of images in each 10° × 10° grid, and the darker the color is, the more remote sensing images there are in that area. The required remote sensing images were selected through conditional queries and downloaded in batches using the Machine-to-Machine (M2M) API.

 doi:10.20944/preprints202305.0134.v1

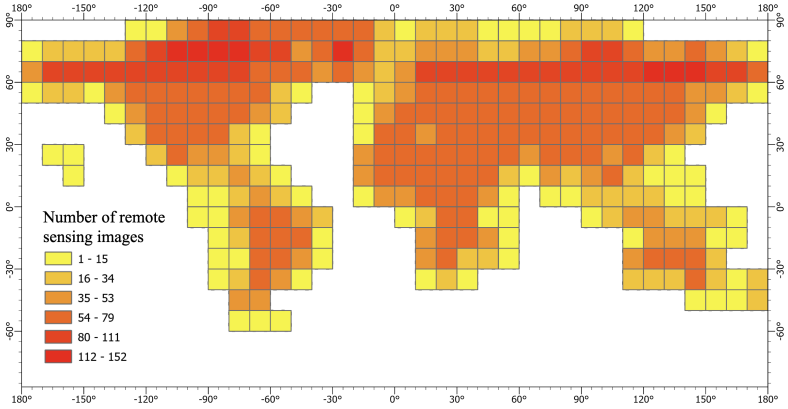


Figure 1. The quantity and distribution of remote sensing image data.

2.1.2. GDEM v3

The DEM data is the third version of The Advanced Spaceborne Thermal Emission and Reflection Radiometer (ASTER) Global Digital Elevation Model (GDEM) program[37], with a resolution of 30 meters. It was jointly developed by the Ministry of Economy, Trade and Industry (METI) of Japan and NASA of the United States, and is available for free download and use. Three versions of data were released in 2009, 2011, and 2019 respectively.

2.1.3. GLC products

FROM_GLC [38] is a global 30m land cover product produced by Gong et al. using Landsat Thematic Mapper (TM) and Enhanced Thematic Mapper Plus (ETM+) data. It classified over 6600 Landsat TM data scenes after 2006 and over 2300 Landsat TM and ETM+ data scenes before 2006, all selected from the green season. GLC_FCS [15] is produced by Zhang and Liu et al., is the first global land cover dataset to provide a refined classification system, including 16 global LCCS land cover types and 14 detailed and regional land cover types, with high classification accuracy at 30m resolution. The accuracy of FROM_GLC in 2015 ranges from 57.71% to 80.36%, while the accuracy of GLC_FCS30 in 2015 ranges from 65.59% to 84.33%.

2.2. Framework Design

HALF provides solutions for each stage of land cover mapping, as shown in Figure 2. It takes remote sensing image data, auxiliary data, and prior GLC products as input. Initially, a large number of initial sample points are automatically generated. Then, training features are obtained by overlaying the sample points with remote sensing and auxiliary data, and the classifier model is trained. Finally, the final land cover product is obtained by mosaicking the classification results of remote sensing images. HALF accelerates the execution speed of the model in the mapping process through Spark, encapsulates the models for each stage using Docker, organizes the process using the Airflow workflow engine, reads and processes remote sensing image data using GDAL, and stores metadata information and performs spatial operator operations using Postgres SQL. HALF's mapping process mainly includes the following key stages.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 May 2023 doi:10.20944/preprints202305.0134.v1

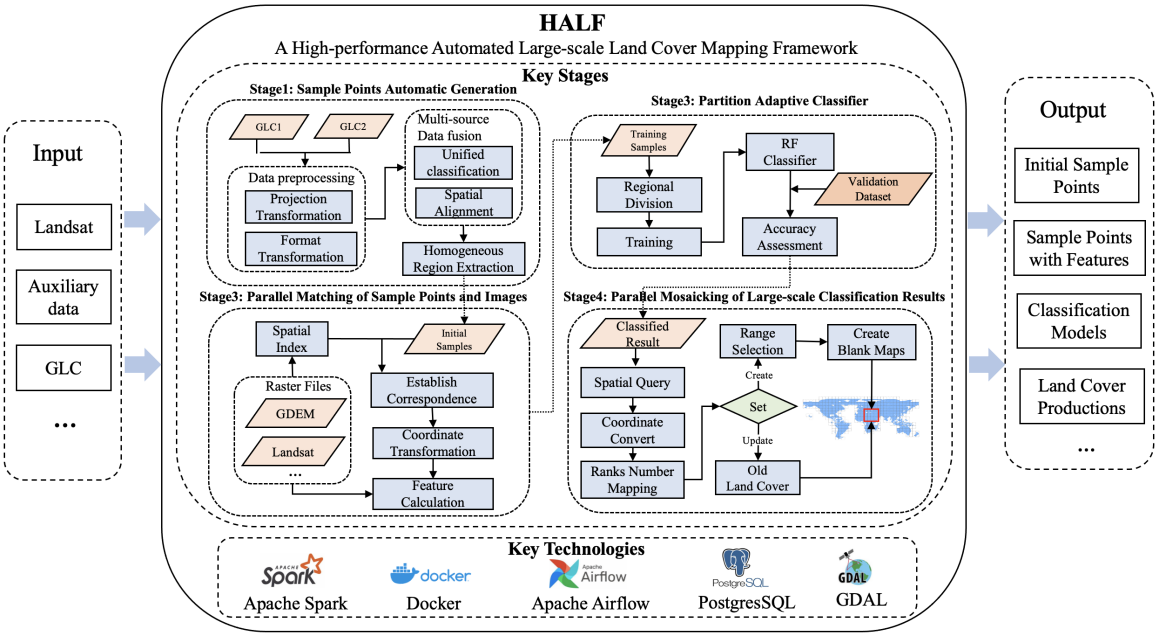


Figure 2. Framework of HALF.

2.2.1. Stage 1: Sample Points Automatic Generation

Collecting samples is one of the most important factors in land cover classification problems, and it can have a significant impact on classification results, sometimes even greater than the impact of the classifier model [39]. The collection of sample data can be categorized into manual interpretation and derivation of training samples from existing land cover products. Many researchers have proven the benefits of exploring existing classification knowledge from available products [40–43]. The method is capable of producing a substantial amount of spatial samples that are evenly distributed without any manual intervention.

Several land cover products have been created based on remote sensing imagery using different classification systems. However, there is no universally accepted system among researchers and the international community. Table 1 lists the existing GLC classification systems. Although the classification systems constructed by different products and organizations are not exactly the same, there are certain similarities in category delineation. Therefore, it is feasible to synthesize and generate new samples from existing products. GLC_FCS classifies all land cover into nine primary classes, including Cropland, Forest, Grassland, Shrubland, Wetlands, Water, Impervious surfaces, Bare areas, and Ice and snow. FROM_GLC classifies all land cover into 11 primary classes, which includes the nine primary classes mentioned before (Cropland, Forest, Grassland, Shrubland, Wetlands, Water, Impervious surfaces, Bare areas, and Ice and snow), with the additional classes of Cloud and Tundra.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 May 2023 doi:10.20944/preprints202305.0134.v1

Table 1. Comparison of Existing GLC Classification Systems.

USGS 1972	CORINE 1985	FROM_GLC 2013	GlobalLand30 2014	GLC_FCS 2021
Forest	Forest and semi-natural areas	Forest	Forest	Forest
Agricultural	Agricultural areas	Crop	Cultivated land	Cropland
		Shrub	Shrubland	Shrubland
Range		Grass	Grassland	Grassland
Wetlands	Wetlands	Wetland	Wetland	Wetlands
Urban or built-up	Artificial surfaces	Impervious	Artificial surfaces	Impervious surfaces
Barren		Bareland	Bareland and tundra	Bare areas
Water	Water bodies	Water	Water bodies	Water body
Perennial snow and ice		Snow/Ice	Permanent snow/ice	Permanent ice and snow
Tundra		Tundra		
		Cloud		

The essence of sample selection is to obtain the spatial position of a certain pixel and its corresponding land cover type, which can be directly derived from existing products. Selecting high-precision training sample points from land cover products is the key and prerequisite to obtain high-precision classification results. In order to ensure the reliability of samples, this section proposes a method for extracting homogenous areas from multi-source product data. The idea of homogenous area extraction is shown in Figure 3. First, select two land cover products with identical spatial range, resolution, and data acquisition time. Next, search for a (7×7) pixel range with an identical classification label in the same spatial position within a single product and between the two products. . Finally, consider the middle (3×3) area as a homogenous region and quantify the number and distribution of these areas. The central pixel is selected as a high-confidence sample for extraction, thereby eliminating spatial deviations between the two land cover products and improving the reliability of samples.

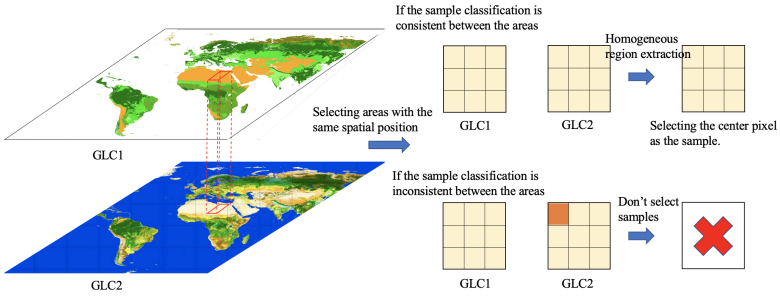


Figure 3. Schematic diagram of homogeneous region extraction.

2.2.2. Stage 2: Parallel Matching of Sample Points and Images

Initial sample points from existing land cover products only includes longitude, latitude, and land cover labels, lacking the features required for classifier training input. It needs to be matched with remote sensing images and auxiliary data to generate final sample points. There is a significant computational burden when matching sample points and remote sensing images. The purpose of the process is to correspond the spatial position of sample points with the pixel position of remote sensing images, in order to read band information and perform calculations. The process of image matching includes: obtaining the image corresponding to the point through a spatial index; converting the longitude and latitude of the sample point to the plane coordinate system of the image, and then transforming it to the pixel row and column position through affine six-parameter transformation; and obtaining the average value in the window as the desired value. In the production process of GLCs, the number of remote sensing images reaches more than

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 May 2023 doi:10.20944/preprints202305.0134.v1

tens of thousands, and the number of spatial points in the training set can reach billions. The data volume of remote sensing images and auxiliary data can reach tens of terabytes. Traditional desktop software and unoptimized matching methods are unable to complete this process, so a more efficient matching method is needed to handle it. HALF proposes a Spark-based sample point-remote sensing image matching method. The key to improving efficiency is to first establish an efficient correspondence relationship between sample points and images through a spatial index, and distribute the reading tasks of different images to each node. For each image, all matching points are read at once, thus avoiding the significant IO pressure caused by frequent image reading. The method is illustrated as follows:

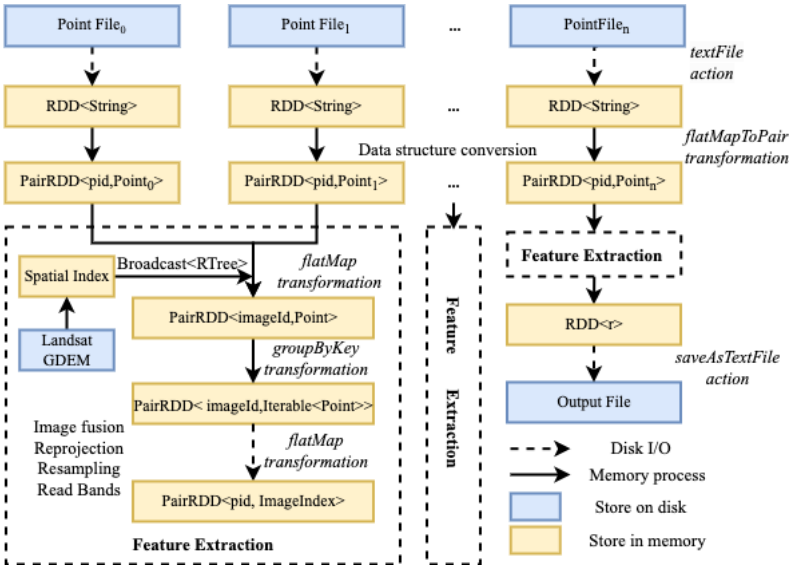


Figure 4. Parallel matching method for sample points and remote sensing images.

- Read all the center point data to be matched into memory, use the longitude and latitude of the point to form a unique identifier pid, read its spatial and attribute information to form a spatial point object, and compose a tuple Tuple<pid, Point>;
- Read the Landsat image metadata information, use the boundary coordinates to construct a spatial index STRtree, and use the Broadcast variable in Spark to transmit it to each node. The Broadcast variable will save a copy on each node, saving the network transmission cost for multiple calls;
- Distribute the computing tasks based on rid. To address the issue of data skew resulting from unequal allocation of images to different computing nodes during task distribution, a custom partitioning strategy is developed, and the groupByKey operator is used to allocate data evenly across all computing nodes as much as possible;
- For each image, parallel operations are performed using GDAL. Image fusion, reprojection and resampling are carried out on the image, and the row and column numbers of the sample point's longitude and latitude on the remote sensing image are calculated. If a sample point is covered by clouds for most of the year or its quality is poor, it is discarded. The feature indicators are calculated for data points that meet the quality criteria, and both the data point and its corresponding match are outputted.

2.2.3. Stage 3: Partition Adaptive Classifier

The adaptive classification model is used to train the data in each region separately. Typically, the entire research area is divided into small areas, and the corresponding sample data is used to train the local adaptive model in each area. The classification results from all areas are merged to generate the final experimental outcomes. Although this approach may not be as straightforward in concept and implementation as global modeling, it overcomes the issue of overfitting due to small areas or underfitting due to large areas by sampling

 doi:10.20944/preprints202305.0134.v1

and training based on the unique characteristics of each region. Hankui [10] quantitatively compared the effectiveness of global modeling and adaptive modeling in the application of land cover in the United States, and their findings demonstrated that local modeling achieved greater classification accuracy than global modeling. In this study, the research area is segmented into a 10°×10° grid.

The classifier provided by HALF is the random forest model. Gómez et al. [44] compared various classification models, such as artificial neural networks, decision trees, support vector machines, random forests, and clustering, in the mapping of land cover, and found that the random forest algorithm has the advantages of insensitivity to noise, good anti-overfitting performance, and good robustness to data reduction. In addition, it is relatively simple to set the parameters, as only the number of trees and the number of features for decision tree branching need to be set. It is suitable for training data with different distribution features in multiple different partitions.

2.2.4. Stage 4: Parallel Mosaicking of Large-scale Classification Results

The size of a Landsat satellite image is 185 km x 185 km. Therefore, when producing large-scale land cover products, multiple scenes need to be stitched together. The main purpose of image stitching is to convert the trained remote sensing images to the map and output the final classification results. In this process, there are a huge number of images. If the images are stitched in serial order, the computational pressure will be enormous. Thus, a certain strategy needs to be adopted to solve the problem of low efficiency in the image merging process.

This process involves coordinate system transformation, image registration, and overlap area processing. Traditional image mosaicking methods first specify the coordinate system of one image as the reference, and then register other images with it to determine the attribute values of corresponding coordinate points. If images are processed sequentially in this process, the computational efficiency is slow. HALF proposed a image mosaicking method and Figure 5 shows its schematic diagram. Traditional image processing operates on a per-image basis, and parallelism is not sufficient. In this method, the globe is divided into a grid of 10°×10° cells. The spatial index is used to query the remote sensing images corresponding to each cell. Each cell is then subdivided into several smaller secondary images, with a specified resolution and edge length. Each cell is divided into m×n sub-images. After obtaining the spatial information of the sub-images, a blank image slice file corresponding to each sub-image is created through GDAL. The spatial relationship between each sub-image and the remote sensing image is established, and the row and column numbers on the remote sensing image are transformed to the corresponding row and column numbers on the sub-image through affine coordinate transformation. The specific values are written to the image slice file, and finally, all sub-images are merged into a complete classification result through the gdalbuildvrt and gdal_translate tools provided by GDAL. The spatial resolution of the customized product can also be used to complete the resampling work during the mosaicking process.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 May 2023 doi:10.20944/preprints202305.0134.v1

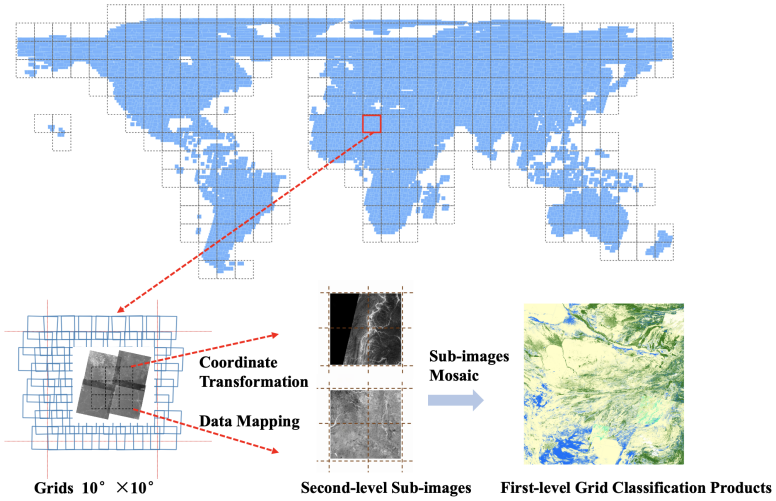


Figure 5. Schematic diagram of the classification result mosaic. The world is partitioned into a grid system with each grid spanning 10° × 10°, and each grid is further divided into second-level sub-images. The blue border indicates the extent of the Landsat image, and when there are classification results within the grid, the pixels of the second-level sub-images within the spatial extent of the classification result are updated, and the final mosaic product is generated.

The parallel mosaicking method is demonstrated in Figure 6. The method first establishes the necessary parameters and determines the relationship between the sub-images and the remote sensing images. It then distributes tasks by using the sub-images as the parallel unit. When a sub-image corresponds to multiple remote sensing images, first determine whether it is a no-data value, otherwise there may be a scenario where the invalid value will overwrite the valid value, and determine whether to output the latest results or all results based on the established rules. If the program is set to output multiple results, a blank view frame for multiple times is created to fill the corresponding sub-image with the corresponding results.

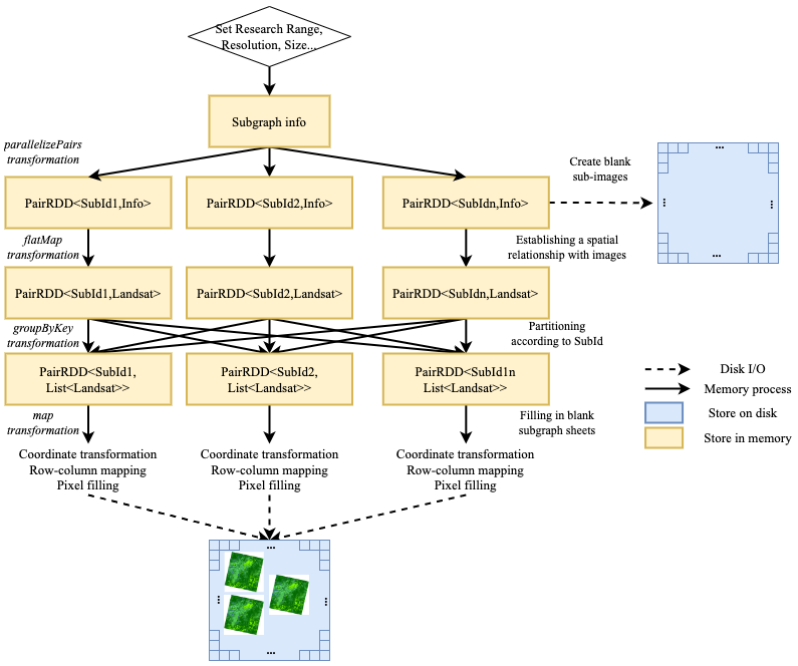


Figure 6. Large-scale image mosaicking method.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 May 2023 doi:10.20944/preprints202305.0134.v1

- Set the spatial extent, resolution, number of bands, and other parameters for different application scenarios. Then create blank map sheets. For example, a latitude and longitude range of 10°×10° corresponds to a grid of geographic extent of approximately 1100 km in length and width. If the sub-image resolution is set to 100 m, and each sub-image size is 256 × 256 pixels, then there are approximately 44×44 sub-images in this spatial range. The metadata information of the sub-images is stored in binary format, including spatial location and description, and files are created based on whether they already exist.;
- Read the image information, and use the flatMap operator to match the spatial position of the sub-image with the metadata of the remote sensing image to establish a spatial correspondence. In this step, the input is PairRDD<SubId, RasterInfo>, and after conversion, the output is PairRDD<SubId, image> which describe the sub-images and the image;
- Use the unique identification code SubId of the sub-image as the key to call the groupByKey operator to obtain the PairRDD<SubId, Iterable<image> > and perform task distribution. Each task unit performs the sub-image filling task;
- In each task computing unit, perform coordinate transformation and row/column calculation for the image classification result, and fill in the corresponding pixel points on the sub-image. If no additional settings are made, the result from the latest remote sensing image corresponding to the sub-image is selected for writing.

2.3. HALF Workflow

HALF provides workflow modules that encapsulate the key stages above through containers and organize each link using the workflow engine. The workflow model involves organizing remote sensing data, sample data and processing functions into a structured and configurable set of steps. This approach enables practical problems to be resolved through automation or semi-automation. To encapsulate the model, the required basic environment image is searched and downloaded from the Docker Hub repository. Then, the necessary runtime environment is added, and the docker commit command is used to create a new container image for storage. Relevant parameters of each model are recorded in XML description documents, while the basic model information is described in metadata files during registration. When adding a model, key details such as the model name, description, type, input/output, executable file path, dependent image, add time, user name, and test case are set, as shown in Table 2. Metadata information of the operator is added to the database to complete the final model information storage and registration.

Table 2. Model metadata information.

Column Name	Data Type	Length
ARTIFACT_ID	varchar	50
NAME	varchar	50
DESCRIPTION	varchar	255
USAGES	varchar	100
MAIN_CLASS	varchar	100
CREATE_DATE	datetime	/
VERSION_ID	int	/
KEYWORDS	varchar	150
INPUT	longtext	/
OUTPUT	longtext	/
PARAMETERS	longtext	/
MODEL_PATH	varchar	255
MODIFY_DATE	date	/
TEST_CASE	longtext	/

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 May 2023 doi:10.20944/preprints202305.0134.v1

When organizing the workflow, use the Docker execution command as the Airflow workflow execution statement, access the database configured by Airflow, and monitor the running status of workflow nodes and instances. Airflow [45] is a lightweight workflow manager developed by AirBnB, Inc. Airflow provides a broad range of tools for managing the execution of workflows, such as pausing and resuming workflow execution, stopping and restarting individual workflow steps, and restarting workflows from a certain step. Airflow treats each workflow as a directed acyclic graph (DAG) of tasks with directional, non-cyclic dependencies. The DAG describes the relationships between tasks and defines their execution order. The Apache Foundation lists over 300 existing workflow languages and systems [46], and commonly used workflow systems in different fields have defined various organizational standards [47]. The international academic community has proposed the FAIR principles (Findable, Accessible, Interoperable, Reusable) for open and shared scientific data [48]. To this end, researchers have proposed the Common Workflow Language (CWL) [49], a specification and standard for describing the creation of workflows that provides a universal standard for addressing the main issues in sharing workflows. CWL describes the input and output of workflows in JSON or YAML format and has been applied in fields such as bioinformatics, hydrology, geospatial analysis, and physics. CWL-Airflow is one of the first workflow managers to support the CWL standard. It creates CWL-DAG instances based on workflow structure and executes them, saving the results to an output folder.

The complete mapping process is presented in Figure 7. The research process is divided into several stages, including sample generation, data matching, model training, image mosaic, and data update. The model description file specifies the parameters for each workflow node, and the input and output nodes for each workflow are designed accordingly.

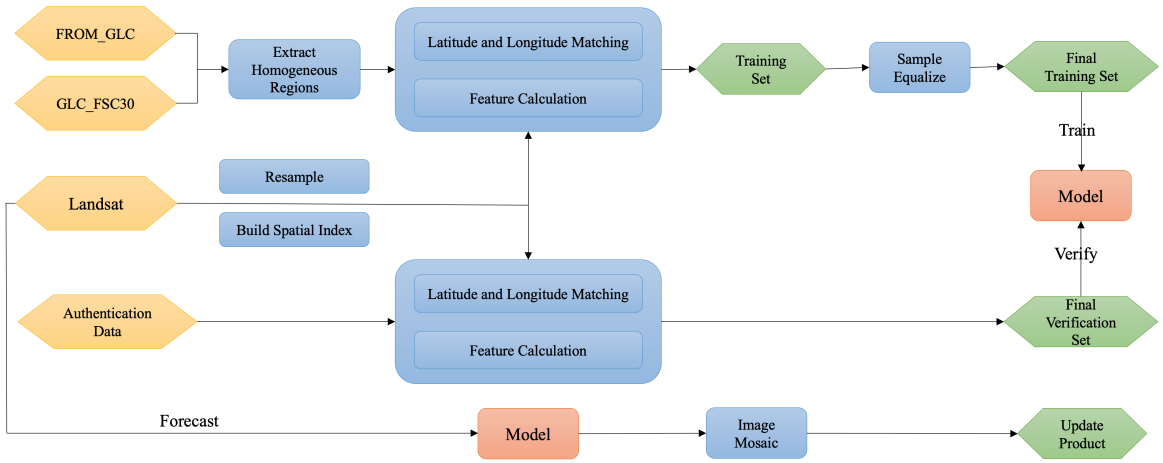


Figure 7. Conceptual design of the workflow. Describes the cartographic phase of HALF, with yellow nodes representing external data, green nodes representing computation results, blue nodes representing models, and red nodes representing classification models.

In HALF, users can establish models by describing the metadata information of workflow nodes in the description file and setting the input and output parameters of the nodes according to different application requirements. HALF quickly builds workflows according to different application requirements to achieve customized management and monitoring. For the combined workflow model, the system provides a save function that records node parameter information and spatial location on the interface. After entering the workflow name and comments, the workflow model can be saved to the server for future use. And it allows users to monitor the running status of each step, with multiple states such as success, running, waiting, and failure. Set the input and output of each node, the input can

 doi:10.20944/preprints202305.0134.v1

select the output of the previous node, and the input correction is carried out in this link to ensure that the output format of the previous node is filled in correctly.

2.4. Experimental Environment

The experimental environment consisted of a distributed cluster composed of three Dell Power Edge R720 servers, each with an Intel Xeon E5-2630 v2 2.60 GHz CPU and 30GB of RAM. The three machines shared a NAS storage of 20T. The experimental software and operating environment were Hadoop 2.7.6, Spark 2.3.4, Scala 2.11.8, JDK 1.8, Python 3.8.3 and GDAL 3.0.3.

The research creates container images for the models. Table 6 shows some of the images used for the runtime environment. The initial images are obtained from Docker Hub and further encapsulated to ensure that each model is portable and reusable.

Table 3. Images environment configuration.

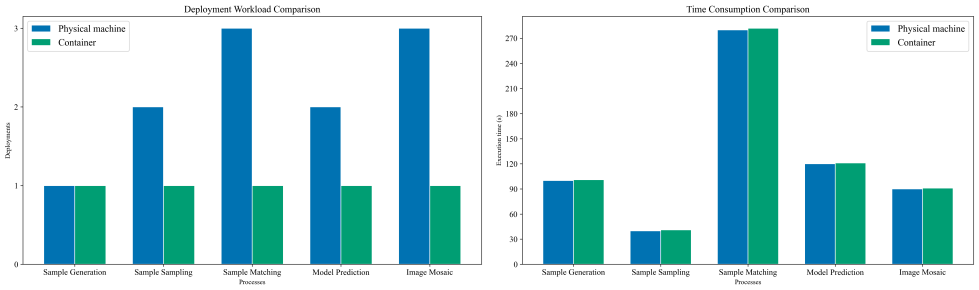
Images	Description	Base images	Models
landuse-py	Used to run models written inPython	python:3.8.3	Samples Generation Classifier Training Image Classification Prediction
gdal-spark	Used to run models written in Java Spark	osgeo/gdal:ubuntu-full-3.6.3	Sample-Image Matching Image Mosaicking
pg12-citus-postgis	Used to storage metadata and spatial queries	postgres:12.14-alpine3.17	Building Spatial Index Spatial Querying
end-point	Used to launch the back-end project and provide network services	java:openjdk-8	Back-end Project

3. Results

3.1. Model Performance Testing

Compare the deployment workload and execution time of operators running tasks in containers and hosts. Deploying conventional applications on a physical machine may depend on multiple development and runtime environments, and the complexity of configuration depends on environment dependencies. For example, generating sample points requires only a python basic environment and GDAL to read images; the sample point matching model is written in Java and also depends on Spark and GDAL; image training requires GDAL under the python environment to fuse and resample images. The deployment of gdal under the python environment on the physical machine can be installed through conda, but the GDAL tool under the Java environment involves gcc compilation, which is extremely cumbersome. By pulling images provided on Docker Hub, the existing environment can be directly pulled and run, and the deployment workload is only one. Therefore, compared with the physical machine environment, the deployment efficiency can be effectively improved in the Docker container environment. Figure 8 shows the total time spent in production. The reason why the running time is longer in the Docker environment than in the physical machine environment is that the startup and destruction of Docker containers will take some time, but the impact is very small in seconds. In addition, Docker containers themselves have the characteristic of lightweight and will not occupy additional memory resources. In summary, deploying the model through Docker does not bring any losses in execution performance, but compared to deploying the environment on a physical machine, it can reduce the configuration workload, making it convenient to migrate and reuse programs when deploying the production environment on new machines.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 May 2023 doi:10.20944/preprints202305.0134.v1



(a) Comparison of time consumption **(b)** Comparison of deployment

Figure 8. Performance comparison between physical machine and container. (a) shows the comparison of time consumption, with various models on the horizontal axis and their corresponding running time on the vertical axis. (b) shows the comparison of deployment amount, with various models on the horizontal axis and an abstract estimate of deployment workload on the vertical axis.

3.2. Automatically Generated Sample Points

Generating training samples based on existing land cover products is capable of obtaining uniformly distributed samples at a large scale. The generated samples are visualized for several regions globally in Figure 9. The method guarantees complete automation of the entire classification process without any human intervention and ensures a large sample size and relatively uniform distribution in the sample space. The classification system of sample points is inherited from prior products, freeing itself from the expert knowledge during the sampling process.

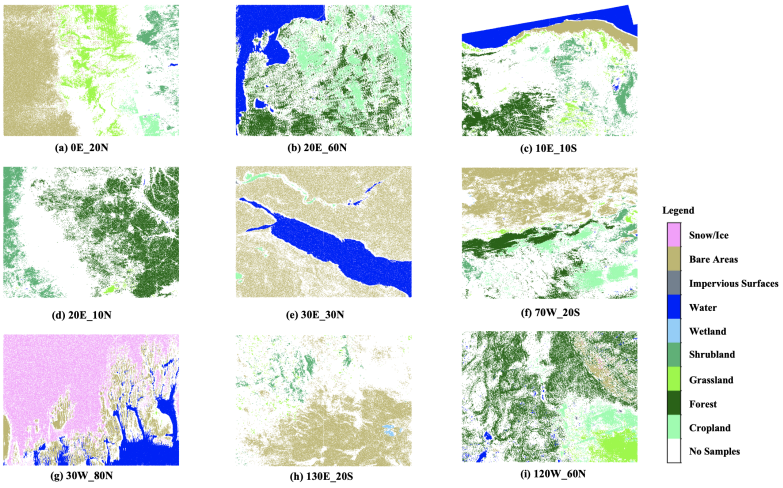


Figure 9. Automatically generated samples. The title of each subfigure indicates the latitude and longitude of the lower-left corner of the region. The legend is shown on the right side of each figure, and the white color indicates that there are no samples in this area.

After matching, the resulting sample points have the characteristics of the final generated dataset as shown in Table 4, which includes 7 spectral features (bands), 4 remote sensing index features - Normalized Difference Vegetation Index (NDVI), Normalized Difference Water Index (NDWI), Enhanced Vegetation Index (EVI), Normalized Burn Ratio (NBR), the latitude and longitude of the center point, the acquisition time of the image, as well as auxiliary indicators such as DEM, slope, and aspect.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 May 2023 doi:10.20944/preprints202305.0134.v1

Table 4. Descriptions of sample points.

Feature	Data Source	Characteristic
Spatial Characteristics	Landsat	Longitude and Latitude
Temporal Characteristics		Image Acquisition Time
Spectral Characteristics		Band1,Band2...Band7
RS Index		NDVI, NDWI, EVI, NBR
Topographic Feature	DEM	DEM, Slope, Aspect
Land Cover Type	GLC	First-level Type

3.3. Performance of Sample Points and Images Matching

In this section, we validated the method and conducted performance tests on different numbers of samples and remote sensing images.The experimental regions are shown in Figure 10, which includes parts of Australia, Africa, and America. The longitude and latitude ranges for the Australian region are (110°E 40°S, 160°E 10°S), for the African region are (20°E 30°S, 50°E 10°N), and for the American region are (110°W 10°N, 50°W 40°N) and (80°W 50°S, 60°W 0°N). The method is able to complete the matching task within a short period of time when dealing with millions of sample points and remote sensing images to be matched within a region. The execution time increases with the increase of sample points and remote sensing images to be matched within a region.

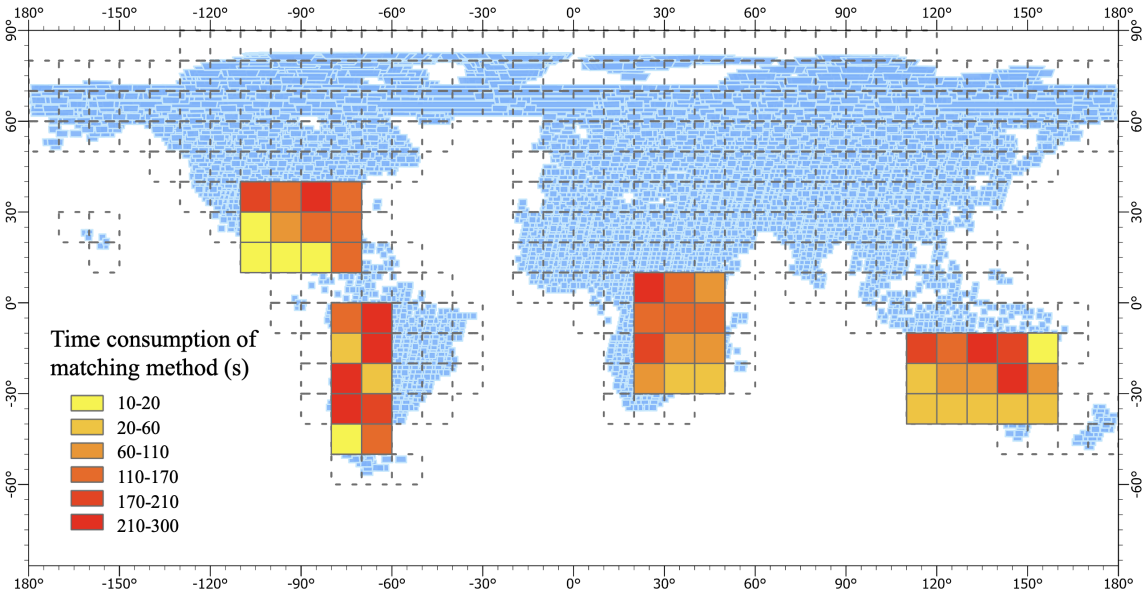


Figure 10. Time consumption statistics of the matching method.

To further validate the performance of the proposed method, the HALF matching method was set to Spark's local computing mode and compared with the method based on GDAL under the same computing resources. The experimental task was to perform sample point-image matching on randomly selected 10,000 to 500,000 points from 131 images and calculate the corresponding total time. Figure 11 shows that the proposed method not only has a significant advantage in running speed, but also is not affected by the sharp increase in point data volume in terms of running time stability. This is because the spatial relationship between points and images is first established using spatial indexing, as the number of points increases, the IO overhead of remote sensing images is relatively fixed.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 May 2023 doi:10.20944/preprints202305.0134.v1

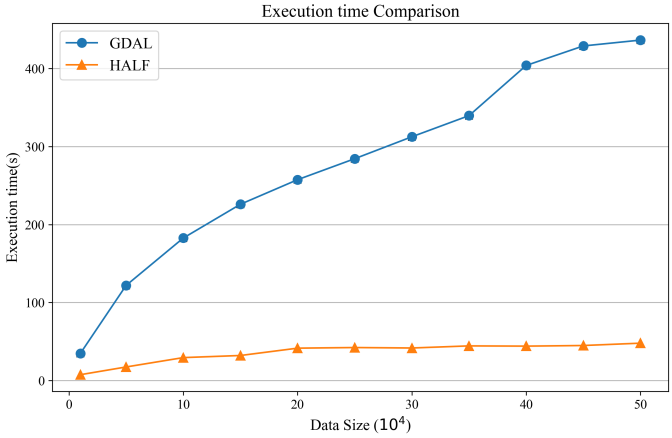


Figure 11. Performance comparison of matching method, the orange line indicates the time consumption of the matching method of HALF, and the blue line indicates the time consumption of the matching method of GDAL.

In order to further investigate the performance of each step of the method, Table 5 provides a detailed breakdown of the time taken for each stage of the two methods at sample sizes of 100,000, 200,000, and 300,000. The results show that the HALF matching method takes significantly less time than the GDAL-based method in the stages of reading points and establishing spatial relationships, as well as reading image values and performing calculations, demonstrating superior computational performance. At the same time, for different sample sizes, the time taken for the Spark-based matching method to distribute tasks to computing nodes has increased, while the time taken to read image values and perform calculations has remained relatively stable. This indicates that the custom partitioning strategy developed by the method effectively addresses the problem of uneven distribution of node images during task distribution, ensuring that the task completion time of each node is as close as possible, maximizing the use of computing resources and reducing computational time.

Table 5. Time consumption analysis for each step of the matching method.

Stage	Data size	100000		200000		300000	
	Time cost(s)	GDAL	HALF	GDAL	HALF	GDAL	HALF
1	Build Spatial Index	0.8	0.8	0.8	0.8	0.8	0.8
2	Read data and create spatial relationships	3.7	2.4	10.5	3.6	26.6	4.4
3	Distribute tasks	/	3.9	/	7.0	/	8.4
4	Compute feature values	179	23.0	247	28.2	283.6	30.7
5	Total	183.5	30.1	258.3	39.7	311	44.3

3.4. Performance of Big-scale Classification Result Mosaicking

Exploration on the performance of the proposed large-scale image parallel mosaic method was conducted by selecting some 10°×10° grids globally for experiments. The experimental regions are shown in Figure 12, which is same as Figure 10. Furthermore, a thematic map was constructed based on the matching time of each grid area, where darker colors indicate longer algorithmic processing time.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 May 2023 doi:10.20944/preprints202305.0134.v1

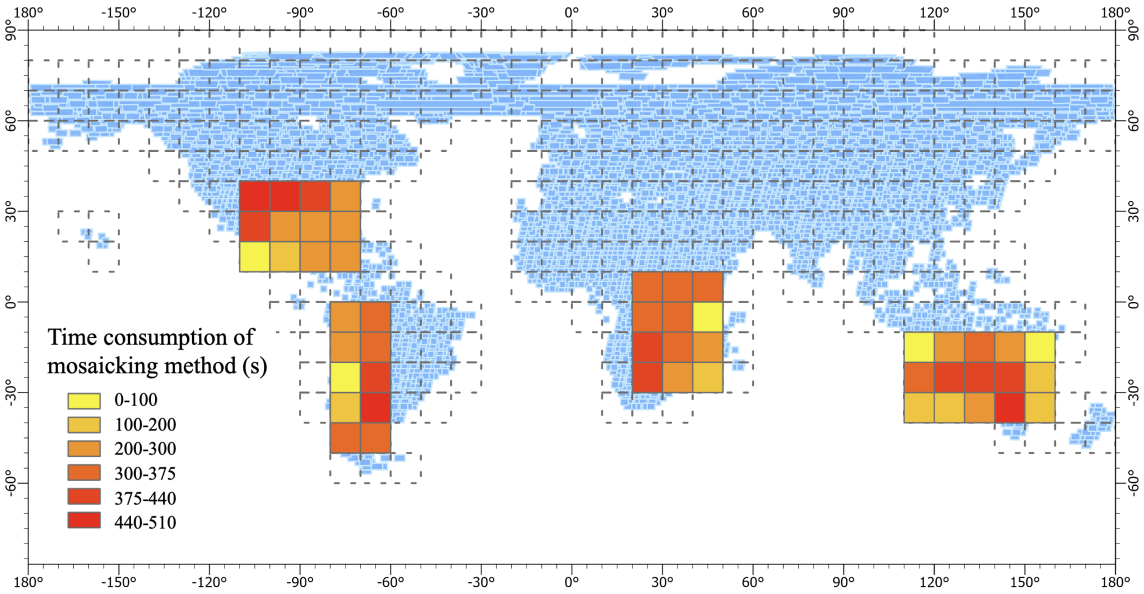


Figure 12. Time consumption of the mosaicking method.

The detailed analysis of the method performance in terms of processing time for the mosaic results of selected regions is presented. It is calculated that the average processing time for one image is around 6.5 seconds. For regions with around 60 images, such as regions 4 and 5, the calculation can be completed in around 400 seconds, while for regions with around 30 images, such as region 11, it takes over 200 seconds. As shown in the experimental results, with the increase of the number of images, the processing time of the mosaic method also gradually increases. Region 7 requires only a little over 2 seconds to process one image on average because only some parts of the region are land, and the sub-images not covered by any input images are not involved in the calculation. Among the selected regions, region 2 has the longest processing time, which is around 17% longer than that of region 4 and 5 with the same number of images. This is because Landsat images have deformation problems in different latitude regions, with larger image sizes in higher latitude regions, leading to longer processing time in high latitude regions. Further experiments are conducted to investigate the performance of the method under different regions and data volumes.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 May 2023 doi:10.20944/preprints202305.0134.v1

Table 6. Data volume and time consumption for partial regions.

Region	Spatial extent	Number of Images	Number of samples	Mosaic Time(s)	Match Time(s)
Region1	(50°S-40°S, 70°W-60°W)	43	9276097	361	166.12
Region2	(40°S-30°S, 70°W-60°W)	64	7557768	481	204.07
Region3	(40°S-30°S, 140°E-150°E)	59	6770182	468	31.20
Region4	(30°S-20°S, 20°E-30°E)	64	3663073	413	99.72
Region5	(30°S-20°S, 140°E-150°E)	64	5716990	408	233.32
Region6	(30°S-20°S, 130°E-140°E)	63	6657017	417	109.19
Region7	(20°S-10°S, 110°E-120°E)	7	6698112	16	175.16
Region8	(20°S-10°S, 20°E-30°E)	63	1675984	401	182.95
Region9	(20°S-10°S, 30°E-40°E)	55	4159308	324	80.81
Region10	(-10°S-0°N, 40°E-50°E)	13	11522471	72	169.84
Region11	(30°N-40°N, 80°W-70°W)	31	16008303	222	143.02
Region12	(0°N-10°N, 30°E-40°E)	62	1022583	359	119.86
Region13	(10°N-20°N, 80°W-70°W)	35	9698301	250	134.51
Region14	(20°N-30°N, 90°W-80°W)	39	12549213	253	146.13
Region15	(20°N-30°N, 80°W-70°W)	34	11688087	224	153.62
Region16	(30°N-40°N, 90°W-80°W)	63	12793479	439	259.34

The study selected five regions with different latitudes and all located on land. Figure 13 shows the changes in method runtime as the number of remote sensing images increases. To facilitate comparison, all these regions are located entirely on land. It was found that regardless of the region, the method's processing time shows a linear trend as the number of images increases. This is because during the calculation, a correspondence is established between the sub-images and the remote sensing images, and the calculation tasks are distributed based on the sub-images. For each calculation task, the remote sensing images are traversed and calculated. Therefore, as the number of images increases, the method's processing time increases almost linearly. In addition, the higher the latitude of the region, the more time it takes to process the same number of images, because Landsat images at higher latitudes undergo distortion due to different projections, resulting in larger image sizes and longer read times.

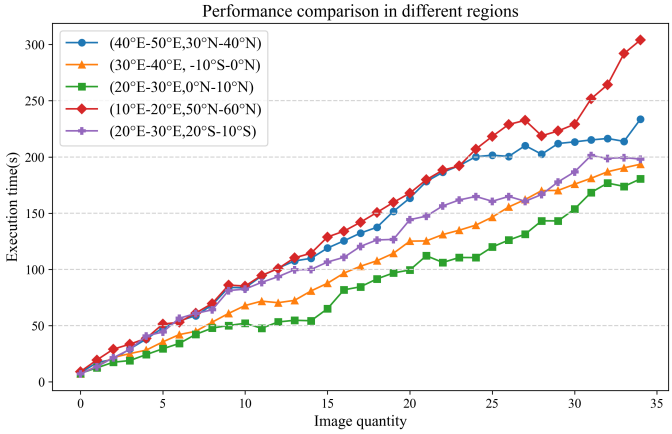


Figure 13. Performance analysis of mosaicking methods.

3.5. Result of Mapping

Figure 14 shows some regions from around the to present the classification results of the images. The results of the high-performance mosaicking method proposed in HALF are demonstrated by stitching together the classification results from regions such as Australia,

 doi:10.20944/preprints202305.0134.v1

Africa, and the Americas. The approach proposed in this study is effective for land mapping and can provide technical support to automate global land mapping.

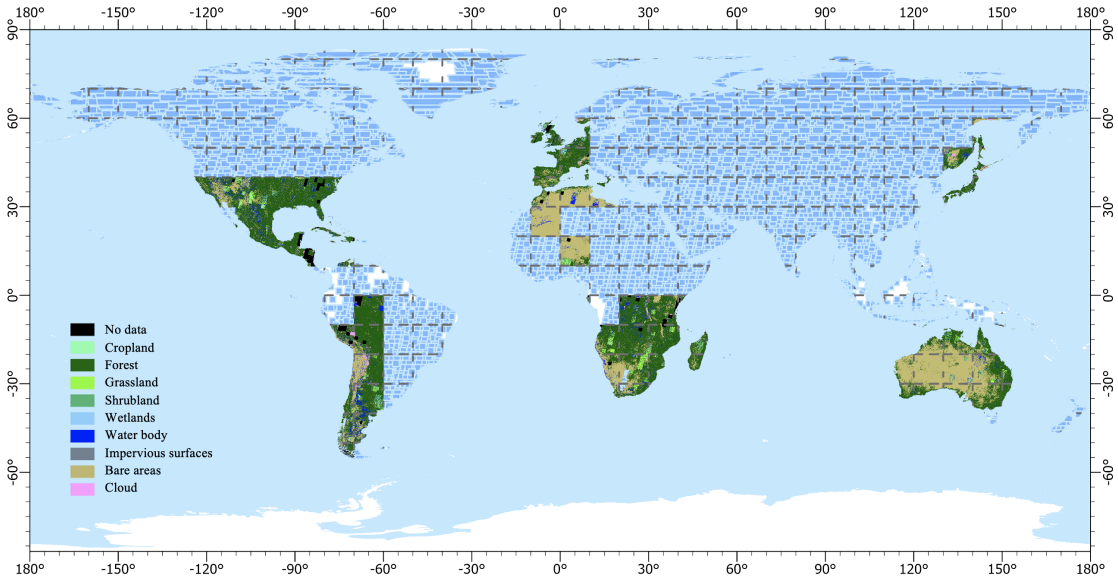


Figure 14. Large-scale classification results mosaic.

Updating classification results at the regional level can be performed on new incoming remote sensing images based on the results of the large-scale mosaic method. As shown in Figure 15, there were missing data in some regions in the initial mapping result for the Korean Peninsula region. By querying the corresponding image row and column numbers based on spatial location information, new images can be inputted to fill in the missing areas. Figure 16 shows the process of updating existing products in the Japan region. In some areas, the product was obscured by clouds, but higher-quality images can be inputted to update the obscured areas.

Figure 15. Update on regional classification results of the Korean Peninsula.

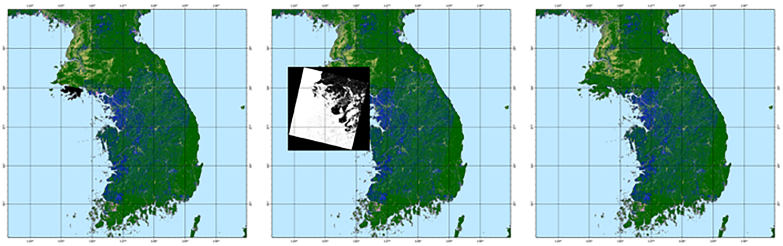


Figure 16. Update on regional classification results of Japan.

 doi:10.20944/preprints202305.0134.v1

4. Discussion

The HALF framework presented in this study offers an automated and high-performance solution for large-scale land cover mapping. By encapsulating each process's models using container technology, HALF addresses the issue of model heterogeneity between different processes, resulting in a significant reduction of deployment workload without sacrificing operational performance. HALF integrated various stages in land cover mapping by using the CWL-Airflow workflow to organize and arrange the models, increasing the automation and flexibility. While the proposed method was experimentally verified for large-scale and efficient mapping, there is a need for service publishing and sharing functions for workflows. Collaborative efforts between experts from various fields and countries are crucial for effective GLC mapping. Real-time sharing of models, processes, and data can have significant implications for land cover mapping.

HALF provides a method of extracting a large number of homogeneous samples from multiple sources of prior products. This method facilitates the generation of a large number of spatially distributed and evenly balanced sample points, significantly reducing the human and material resources utilized in the sample selection process. However, the accuracy of the samples generated by this method is affected by the prior products. In this study, many samples were generated by overlaying classification products automatically, and the first-level classes in the FROM_GLC and GLC_FCS products were extracted. The classification system chosen was relatively coarse, and the samples produced by this method were constrained by the accuracy of the prior products, thereby affecting the training of the classification model. In future research, the quality of the produced samples can be more rigorously controlled by adopting a more refined classification system. In addition, techniques like change detection methods and transfer learning can be utilized to create a substantial number of uniformly distributed and high-precision samples.

HALF optimized and accelerated sample feature matching and classification result mosaic using high-performance computing technology. The specific effects of the method and its performance under different data volumes were discussed and explained in sections 3.4 and 3.5. The experimental results show that the matching method proposed in this study can quickly establish spatial relationships between points and images, and read image attribute data in parallel, effectively improving the computational efficiency of sample-image matching and feature extraction. Its efficiency is more than 10 times faster than conventional matching methods, and it performs well and stably under different data volume scenarios, making it applicable to other remote sensing applications that require large-scale vector and raster data matching. The classification result mosaic method proposed in the study uses image slices as basic units, achieving real-time mapping of large-scale remote sensing image classification results to products and improving mapping efficiency. The total time required for resampling and mosaic of a single Landsat image in a 10°×10° grid was about 6.5 seconds on average, solving the problem of low computational efficiency when synthesizing multiple images. However, there may be practical difficulties in expanding the study area to the entire world, and the computational burden will increase exponentially with increasing research scope and resolution. Therefore, data fusion with other spatiotemporal resolution remote sensing data can be utilized to generate more real-time products if a higher update frequency is desired.

5. Conclusions

The production process of large-scale land cover products is complex and requires more automated tools and higher-performance computing methods to meet the increasing requirements for satellite remote sensing data resolution and product update frequency. HALF aims to improve the operational efficiency of each link in the traditional process of land cover mapping in big data by utilizing high-performance computing technology. To address the heterogeneity of operating systems, running environments, and programming languages between different links, container technology is used to encapsulate models such as sample point generation, sample point-remote sensing image matching, model

training and prediction, and classification result mosaic, supporting model reuse and data sharing, which greatly reduces the workload of deployment.Additionally, a general workflow language is introduced to model the land cover mapping process, organize data and models for each link, and decouple the production model from the overall process, thereby enhancing the automation and flexibility of the mapping process. In the future, we will further explore how to obtain higher-quality sample points from multiple land cover products, encapsulate them into online data services with network technology, and integrate them into an integrated production portal to better support researchers in the field. By continuously improving HALF, we can enhance the efficiency of large-scale land cover mapping and better support global resource monitoring and sustainable development efforts.

References

1. Sterling, S.M.; Ducharne, A.; Polcher, J. The impact of global land-cover change on the terrestrial water cycle. *Nature Climate Change* **2013**, *3*, 385–390. https://doi.org/10.1038/nclimate1690.
2. Feddema, J.J.; Oleson, K.W.; Bonan, G.B.; Mearns, L.O.; Buja, L.E.; Meehl, G.A.; Washington, W.M. The Importance of Land-Cover Change in Simulating Future Climates. *Science* **2005**, *310*, 1674–1678. https://doi.org/10.1126/science.1118160.
3. Ban, Y.; Gong, P.; Giri, C. Global land cover mapping using Earth observation satellite data: Recent progresses and challenges. *ISPRS Journal of Photogrammetry and Remote Sensing* **2015**, *103*, 1–6. https://doi.org/10.1016/j.isprsjprs.2015.01.001.
4. Brown, C.F.; Brumby, S.P.; Guzder-Williams, B.; Birch, T.; Hyde, S.B.; Mazzariello, J.; Czerwinski, W.; Pasquarella, V.J.; Haertel, R.; Ilyushchenko, S.; et al. Dynamic World, Near real-time global 10m land use land cover mapping. *Scientific Data* **2022**, *9*, 251. https://doi.org/10.1038/s41597-022-01307-4.
5. Yu, L.; Du, Z.; Dong, R.; Zheng, J.; Tu, Y.; Chen, X.; Hao, P.; Zhong, B.; Peng, D.; Zhao, J.; et al. FROM-GLC Plus: toward near real-time and multi-resolution land cover mapping. *GIScience & Remote Sensing* **2022**, *59*, 1026–1047. https://doi.org/10.1080/15481603.2022.2096184.
6. Sulla-Menashe, D.; Gray, J.M.; Abercrombie, S.P.; Friedl, M.A. Hierarchical mapping of annual global land cover 2001 to present: The MODIS Collection 6 Land Cover product. *Remote Sensing of Environment* **2019**, *222*, 183–194. https://doi.org/10.1016/j.rse.2018.12.013.
7. Buchhorn, M.; Lesiv, M.; Tsendbazar, N.E.; Herold, M.; Bertels, L.; Smets, B. Copernicus Global Land Cover Layers—Collection 2. *Remote Sensing* **2020**, *12*. https://doi.org/10.3390/rs12061044.
8. Chen, J.; Chen, J.; Liao, A.; Cao, X.; Chen, L.; Chen, X.; He, C.; Han, G.; Peng, S.; Lu, M.; et al. Global land cover mapping at 30m resolution: A POK-based operational approach. *ISPRS Journal of Photogrammetry and Remote Sensing* **2015**, *103*, 7–27. https://doi.org/10.1016/j.isprsjprs.2014.09.002.
9. Li, X.; Gong, P. An "exclusion-inclusion" framework for extracting human settlements in rapidly developing regions of China from Landsat images. *Remote Sensing of Environment* **2016**, *186*, 286–296. https://doi.org/10.1016/j.rse.2016.08.029.
10. Zhang, H.K.; Roy, D.P. Using the 500m MODIS land cover product to derive a consistent continental scale 30m Landsat land cover classification. *Remote Sensing of Environment* **2017**, *197*, 15–34. https://doi.org/10.1016/j.rse.2017.05.024.
11. Radoux, J.; Lamarche, C.; Bogaert, E.V.; Bontemps, S.; Brockmann, C.; Defourny, P. Automated Training Sample Extraction for Global Land Cover Mapping. *Remote Sensing* **2014**, *6*, 3965–3987. https://doi.org/10.3390/rs6053965.
12. Yu, L.; Wang, J.; Li, X.; Li, C.; Zhao, Y.; Gong, P. A multi-resolution global land cover dataset through multisource data aggregation. *Science China Earth Sciences* **2014**, *57*, 2317–2329. https://doi.org/10.1007/s11430-014-4919-z.
13. Wessels, K.J.; Van den Bergh, F.; Roy, D.P.; Salmon, B.P.; Steenkamp, K.C.; MacAlister, B.; Swanepoel, D.; Jewitt, D. Rapid Land Cover Map Updates Using Change Detection and Robust Random Forest Classifiers. *Remote Sensing* **2016**, *8*. https://doi.org/10.3390/rs8110888.
14. Zhang, X.; Liu, L.; Chen, X.; Xie, S.; Gao, Y. Fine Land-Cover Mapping in China Using Landsat Datacube and an Operational SPECLib-Based Approach. *Remote Sensing* **2019**, *11*. https://doi.org/10.3390/rs11091056.
15. Zhang, X.; Liu, L.; Chen, X.; Gao, Y.; Xie, S.; Mi, J. GLC_FCS30: global land-cover product with fine classification system at 30m using time-series Landsat imagery. *Earth System Science Data* **2021**, *13*, 2753–2776. https://doi.org/10.5194/essd-13-2753-2021.
16. Venter, Z.S.; Barton, D.N.; Chakraborty, T.; Simensen, T.; Singh, G. Global 10 m Land Use Land Cover Datasets: A Comparison of Dynamic World, World Cover and Esri Land Cover. *Remote Sensing* **2022**, *14*. https://doi.org/10.3390/rs14164101.
17. Gong, P.; Yu, L.; Li, C.; Wang, J.; Liang, L.; Li, X.; Ji, L.; Bai, Y.; Cheng, Y.; Zhu, Z. A new research paradigm for global land cover mapping. *Annals of GIS* **2016**, *22*, 87–102. https://doi.org/10.1080/19475683.2016.1164247.
18. Camargo, A.; Schultz, R.R.; Wang, Y.; Fevig, R.A.; He, Q. GPU-CPU implementation for super-resolution mosaicking of Unmanned Aircraft System (UAS) surveillance video. In Proceedings of the 2010 IEEE Southwest Symposium on Image Analysis & Interpretation (SSIAI), 2010, pp. 25–28. https://doi.org/10.1109/SSIAI.2010.5483926.
19. Ma, Y.; Song, J.; Zhang, Z. In-Memory Distributed Mosaicking for Large-Scale Remote Sensing Applications with Geo-Gridded Data Staging on Alluxio. *Remote Sensing* **2022**, *14*. https://doi.org/10.3390/rs14235987.

20. Zhang, J.; Ke, T.; Sun, M. Parallel processing of mass aerial digital images base on cluster computer—The application of parallel computing in aerial digital photogrammetry. *Comput. Eng. Appl.* **2008**, *44*, 12–15. https://doi.org/10.1109/ISIP.2008.114.
21. Chen, C.; Tan, Y.; Li, H.; Gu, H. A Fast and Automatic Parallel Algorithm of Remote Sensing Image Mosaic. *Microelectronics & Computer* **2011**, *28*, 59–62.
22. Ma, Y.; Wang, L.; Zomaya, A.Y.; Chen, D.; Ranjan, R. Task-Tree Based Large-Scale Mosaicking for Massive Remote Sensed Imageries with Dynamic DAG Scheduling. *IEEE Transactions on Parallel and Distributed Systems* **2014**, *25*, 2126–2137. https://doi.org/10.1109/TPDS.2013.272.
23. Jing, W.; Huo, S.; Miao, Q.; Chen, X. A Model of Parallel Mosaicking for Massive Remote Sensing Images Based on Spark. *IEEE Access* **2017**, *5*, 18229–18237. https://doi.org/10.1109/ACCESS.2017.2746098.
24. Rabenseifner, R.; Hager, G.; Jost, G. Hybrid MPI/OpenMP Parallel Programming on Clusters of Multi-Core SMP Nodes. In Proceedings of the 2009 17th Euromicro International Conference on Parallel, Distributed and Network-based Processing, 2009, pp. 427–436. https://doi.org/10.1109/PDP.2009.43.
25. Apache Hadoop. https://hadoop.apache.org. accessed 20 April 2023.
26. Zaharia, M.; Chowdhury, M.; Franklin, M.J.; Shenker, S.; Stoica, I.; et al. Spark: Cluster computing with working sets. *HotCloud* **2010**, *10*, 95.
27. Garland, M.; Le Grand, S.; Nickolls, J.; Anderson, J.; Hardwick, J.; Morton, S.; Phillips, E.; Zhang, Y.; Volkov, V. Parallel Computing Experiences with CUDA. *IEEE Micro* **2008**, *28*, 13–27. https://doi.org/10.1109/MM.2008.57.
28. Eldawy, A.; Mokbel, M. SpatialHadoop: A MapReduce framework for spatial data. In Proceedings of the 2015 IEEE 31st International Conference on Data Engineering, ICDE 2015. IEEE Computer Society, 2015, Proceedings - International Conference on Data Engineering, pp. 1352–1363. https://doi.org/10.1109/ICDE.2015.7113382.
29. Aji, A.; Wang, F.; Vo, H.; Lee, R.; Liu, Q.; Zhang, X.; Saltz, J. Hadoop GIS: A High Performance Spatial Data Warehousing System over Mapreduce. *Proc. VLDB Endow.* **2013**, *6*, 1009–1020. https://doi.org/10.14778/2536222.2536227.
30. Shaikh, S.A.; Mariam, K.; Kitagawa, H.; Kim, K. GeoFlink: A Framework for the Real-time Processing of Spatial Streams. *CoRR* **2020**, *abs/2004.03352*. https://doi.org/10.48550/arXiv.2004.03352.
31. Kopp, S.; Becker, P.; Doshi, A.; Wright, D.J.; Zhang, K.; Xu, H. Achieving the Full Vision of Earth Observation Data Cubes. *Data* **2019**, *4*. https://doi.org/10.3390/data4030094.
32. Nüst, D.; Konkol, M.; Pebesma, E.; Kray, C.; Schutzeichel, M.; Przibytzin, H.; Lorenz, J. Opening the publication process with executable research compendia. *D-Lib Magazine* **2017**, *23*, 451.
33. Wang, B.; Zhang, M.; Huang, Q.; Yue, P. A Container-Based Service Publishing Method for Heterogeneous Geo-processing Operators. *Journal of Geomatics* **2021**, pp. 174–177. https://doi.org/10.14188/j.2095-6045.2020314.
34. Huffman, J.; Forsberg, A.; Loomis, A.; Head, J.; Dickson, J.; Fassett, C. Integrating advanced visualization technology into the planetary Geoscience workflow. *Planetary and Space Science* **2011**, *59*, 1273–1279. https://doi.org/10.1016/j.pss.2010.07.015.
35. Yue, P.; Zhang, M.; Tan, Z. A geoprocessing workflow system for environmental monitoring and integrated modelling. *Environmental Modelling Software* **2015**, *69*, 128–140. https://doi.org/10.1016/j.envsoft.2015.03.017.
36. Chen, Y.; Lin, H.; Xiao, L.; Jing, Q.; You, L.; Ding, Y.; Hu, M.; Devlin, A.T. Versioned geoscientific workflow for the collaborative geo-simulation of human-nature interactions – a case study of global change and human activities. *International Journal of Digital Earth* **2021**, *14*, 510–539. https://doi.org/10.1080/17538947.2020.1849439.
37. Gesch, D.; Oimoen, M.; Danielson, J.; Meyer, D. VALIDATION OF THE ASTER GLOBAL DIGITAL ELEVATION MODEL VERSION 3 OVER THE CONTERMINOUS UNITED STATES. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* **2016**, *XLI-B4*, 143–148. https://doi.org/10.5194/isprs-archives-XLI-B4-143-2016.
38. Gong, P.; Wang, J.; Yu, L.; Zhao, Y.; Zhao, Y.; Liang, L.; Niu, Z.; Huang, X.; Fu, H.; Liu, S.; et al. Finer resolution observation and monitoring of global land cover: first mapping results with Landsat TM and ETM+ data. *International Journal of Remote Sensing* **2013**, *34*, 2607–2654. https://doi.org/10.1080/01431161.2012.748992.
39. Foody, G.M.; Arora, M.K. An evaluation of some factors affecting the accuracy of classification by an artificial neural network. *International Journal of Remote Sensing* **1997**, *18*, 799–810. https://doi.org/10.1080/014311697218764.
40. Du, P.; Lin, C.; Chen, Y.; Wang, X.; Zhang, W.; Guo, S. Training Sample Transfer Learning from Multi-temporal Remote Sensing Images for Dynamic and Intelligent Land Cover Classification. *Journal of Tongji University (Natural Science Edition)* **2022**, *50*, 955–966. https://doi.org/10.11908/j.issn.0253-374x.22200.
41. Huang, Y.; Liao, S. Automatic collection for land cover classification based on multisource datasets. *Journal of Remote Sensing* **2017**, *21*, 757–766. https://doi.org/10.11834/jrs.20186371.
42. Liu, K.; Yang, X.; Zhang, T. Automatic Selection of Clasified Samples with the Help of Previous Land Cover Data. *Journal of Geo-information Science* **2012**, *14*, 507–513. https://doi.org/10.3724/SP.J.1047.2012.00507.
43. Xiaodong, W.T.L.J.X.L.Y.H.S.Z.H. An Automatic Sample Collection Method for Object-oriented Classification of Remotely Sensed Imageries Based on Transfer Learning. *Acta Geodaetica et Cartographica Sinica* **2014**, *43*, 908. https://doi.org/10.13485/j.cnki.11-2089.2014.0163.
44. Gómez, C.; White, J.C.; Wulder, M.A. Optical remotely sensed time series data for land cover classification: A review. *ISPRS Journal of Photogrammetry and Remote Sensing* **2016**, *116*, 55–72. https://doi.org/10.1016/j.isprsjprs.2016.03.008.
45. Apache Airflow. https://airflow.apache.org. accessed 20 April 2023.

 doi:10.20944/preprints202305.0134.v1

46. Amstutz, P.; Mikheev, M.; Crusoe, M.R.; Tijanić, N.; Lampa, S.; et al.. Existing Workflow Systems. https://s.apache.org/existing-workflow-systems. accessed 18 April 2023.
47. Leipzig, J. A review of bioinformatic pipeline frameworks. *Briefings in bioinformatics* **2017**, *18*, 530–536. https://doi.org/10.1093/bib/bbw020.
48. Schultes, E.; Wittenburg, P. FAIR Principles and Digital Objects: Accelerating Convergence on a Data Infrastructure. Springer International Publishing, 2019, pp. 3–16.
49. Common Workflow Language. http://www.commonwl.org. accessed 20 April 2023.