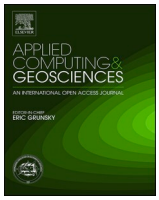
Applied Computing and Geosciences 19 (2023) 100126

Contents lists available at ScienceDirect

Applied Computing and Geosciences

journal homepage: www.sciencedirect.com/journal/applied-computing-and-geosciences



Geoweaver_cwl: Transforming geoweaver AI workflows to common workflow language to extend interoperability

Amruta Kale [a], Ziheng Sun [b,c], Chao Fan [a], Xiaogang Ma [a,d,*]

[a] Department of Computer Science, University of Idaho, Moscow, ID, 83844, USA
[b] Center for Spatial Information Science and Systems, George Mason University, Fairfax, VA, 22030, USA
[c] Department of Geography and Geoinformation Science, George Mason University, Fairfax, VA, 22030, USA
[d] Earth and Planets Laboratory, Carnegie Institution for Science, Washington, DC, 20015, USA

ARTICLE INFO

Keywords:
AI workflows
Explainability
Transparency
Provenance
Common workflow language

ABSTRACT

Recently, workflow management platforms are gaining more attention in the artificial intelligence (AI) community. Traditionally, researchers self-managed their workflows in a manual and tedious way that heavily relies on their memory. Due to the complexity and unpredictability of AI models, they often struggled to track and manage all the data, steps, and history of the workflow. AI workflows are time-consuming, redundant, and error-prone, especially when big data is involved. A common strategy to make these workflows more manageable is to use a workflow management system, and we recommend Geoweaver, an open-source workflow management system that enables users to create, modify and reuse AI workflows all in one place. To make our work in Geoweaver reusable by the other workflow management systems, we created an add-on functionality ***geoweaver_cwl***, a Python package that automatically converts Geoweaver AI workflows into the Common Workflow Language (CWL) format. It will allow researchers to easily share, exchange, modify, reuse, and build a new workflow from existing ones in other CWL-compliant software. A user study was conducted with the existing workflows created by Geoweaver to collect suggestions and fill in the gaps between our package and Geoweaver. The evaluation confirms that ***geoweaver_cwl*** can lead to a well-versed AI process while disclosing opportunities for further extensions. The ***geoweaver_cwl*** package is publicly released online at https://pypi.org/project/geoweaver-cwl/0.0.1/.

1. Introduction

We are witnessing a widespread adoption of artificial intelligence (AI) and machine learning (ML) in our everyday life. The recent success of deep learning (DL) has largely contributed to the huge success of AI/ML models. DL algorithms are widely used in mission-critical applications like healthcare, autonomous robots and vehicles, image classification, and detection. Despite the significant improvement in performance and predictions, the black-box nature of DL algorithms can raise social and ethical questions about their operations and results. Even the programmer designing the complex AI/ML model finds it difficult to gain insight into an internal system that is often opaque. This issue has extended the research focus from improving accuracy to explainable and interpretable ML models (Doshi-Velez et al., 2017; Gilpin et al., 2018; Adadi and Berrada, 2018; Wing, 2020; Sun et al., 2022).

Recent interests in explainable artificial intelligence (XAI) and trustworthy artificial intelligence (TAI) have achieved great momentum in making AI/ML models more explainable, interpretable, and transparent (Adadi and Berrada, 2018; Rudin, 2018, 2019; Wing, 2020). XAI proposes a shift toward more transparent AI. It aims to develop a set of strategies to make ML models more explainable while maintaining their high predictive accuracy (Ribeiro et al., 2016; Gunning and Aha, 2019). As the field of XAI continues to expand, it is important to develop new research strategies that include the provenance of upstream steps and history model runs. The diverse nature of AI/ML models in the field of XAI requires a multi-disciplinary approach, and in our previous papers, we highlighted the importance of provenance documentation and its benefit for AI/ML models (Kale et al., 2023; Kale and Ma, 2023). We suggest that adopting approaches and methods from the field of provenance will help to generate resourceful explanations and improve reproducibility (Ma et al., 2017; Zeng et al., 2019; Kale et al., 2023).

* Corresponding author. Department of Computer Science, University of Idaho, Moscow, ID, 83844, USA.
E-mail address: max@uidaho.edu (X. Ma).

https://doi.org/10.1016/j.acags.2023.100126
Received 19 November 2022; Received in revised form 31 May 2023; Accepted 7 June 2023
Available online 14 June 2023

Provenance provides transparency into the data processing steps, allowing researchers to understand how the data was created and processed. This enables researchers to reproduce the results by repeating the same steps, ensuring that the results are reliable and can be trusted. Additionally, provenance can be utilized for quality control purposes by allowing researchers to detect errors or inconsistencies in the data processing steps. This helps ensure that the data is of superior quality and that the outcomes are trustworthy and reproducible.

Scientific workflow management systems like Kepler (Altintas et al., 2004), DataRobot (DataRobotCloud, 2012), Datatron (Datatron, 2016), Metaclip (Bedia et al., 2019), Amazon SageMaker (Das et al., 2020), and Geoweaver (Sun et al., 2020) are widely utilized for data analysis, providing a means for informed decision-making, and promoting innovation. These tools provide several ways to explore the provenance repository by tracking model activity, recording changes in the data and model, and outlining best practices for data processing. However, their growing popularity has led to concerns regarding collaboration and the possibility of hindering workflow reusability and portability. To address this concern, we support a standardized approach to computational workflows that fosters collaboration and mitigates these risks. This paper highlights the significance of Common Workflow Language (CWL), a practical set of standards that enables the description and sharing of computational workflows among a diverse community of users in various fields of science and engineering.

This paper provides an overview of the CWL standards and our new Python package, ***geoweaver_cwl*** that transform Geoweaver AI/ML workflows into CWL scripts. We emphasize the importance of standardization in promoting collaboration and workflow portability and highlight how CWL can provide a practical solution to these challenges. By promoting the adoption of CWL standards and our ***geoweaver_cwl*** package, we aim to advance the field of computational workflows and promote their effective use in scientific research and engineering. This paper will describe how the tool is developed and implemented in our use cases and is organized as follows. In section 2, we first describe an overview of CWL, followed by the conceptual framework of Geoweaver, and then describe the architecture of ***geoweaver_cwl***. In section 3, we demonstrate our Python package by applying a use case from the Geoweaver platform and assess the quality of the package and its influence in Geoweaver. In section 4, we discuss the importance of adopting CWL standards and highlight the future direction of our work. Finally, we conclude with a few additional remarks.

2. Technical framework of the *geoweaver_cwl* package

2.1. An overview of the common workflow language

CWL is a community standard to describe command-line-based workflows (Amstutz et al., 2016). It offers a typical but simplified set of generalizations that are commonly implemented in many popular-workflow management systems. The language's declarative format enables users to describe the process of executing diverse software tools and workflows through their command-line interface. Previously, to link the command-line tools researchers need to write shell scripts. Although these scripts offer an efficient approach to accessing the tools, writing, and maintaining them requires specialized knowledge. As a result, researchers spend more time maintaining the scripts than conducting their research (Sun et al., 2022). However, with the increase in workflow popularity, the number of workflow management tools has increased, and each of them has its standards for specifying the tools and workflows. This has reduced the portability and interoperability of these workflows. CWL aims to reduce the barrier to researchers using these technologies by providing a standard to unify them. CWL standards explicitly support the usage of container technologies like Docker, Singularity, Shifter. These container technologies allow encapsulation of software dependencies and system configuration ensuring that the workflow can be executed in a consistent and reproducible environment, regardless of the underlying system (Pahl et al., 2017). In addition to providing a consistent execution environment, container technologies also facilitate the sharing and reuse of workflows. By packaging the workflow and its dependencies in a container image, it can be easily shared and executed on other systems without the need to install additional software or configure the environment. Overall, container technologies play a critical role in ensuring the reproducibility and portability of CWL workflows.

2.2. Conceptual framework of the geoweaver workflow management system

Geoweaver is a unique platform designed for NASA's Earth Observing System Data and Information System (EOSDIS), which provides earth scientists with the ability to manage, share, replicate, and reuse artificial intelligence/machine learning (AI/ML) workflows. The platform is equipped with a user-friendly graphical interface that enables individuals with limited programming experience to create and execute workflows with ease. Geoweaver offers a comprehensive range of AI workflows that include data preprocessing, training, testing of AI algorithms, and post-processing of results into an ad hoc automated workflow, which is particularly useful for AI practitioners (Sun et al., 2020).

One of the unique features of Geoweaver is its integration with open-source software tools that are commonly used in geospatial data analysis. This integration enables users to incorporate different software packages into their workflows seamlessly without requiring extensive knowledge of each tool individually. Furthermore, Geoweaver's versatility enables it to support various data formats and processing capabilities, making it a valuable tool for individuals working in different fields such as environmental science, agriculture, and urban planning. Geoweaver's scalability is another notable advantage. The platform is built on a distributed computing architecture that can manage large geospatial datasets and perform computationally intensive analyses. Additionally, Geoweaver supports high-performance computing resources such as multi-core CPUs, clusters, and cloud computing, which enhance its computational power. Geoweaver is a unique and valuable tool for managing geospatial data workflows. Its flexible workflow composition, integration with open-source software tools, scalability, and support for a wide range of data formats and processing capabilities make it an ideal platform for AI practitioners and earth scientists.

The fundamental design of Geoweaver is organized into three modules (Host, Process, and Workflow), which enable AI practitioners to sort and reuse their AI/ML experiments.

- Host: This module serves as the cornerstone for the framework, differentiating it from other workflow management system. It enables users to connect to several resources such as virtual machines, Jupyter server instances, Secure shell (SSH), and third-party computing platforms like Google Earth Engine, Jupyter Notebook Server, and Google Colab. Additionally, the file transfer services (file uploading from local computers to remote servers, and file downloading from remote servers to local computers) provided by the host module allow users to transfer their workflow from one platform to another.
- Process: This module includes five submodules and one database. As most of the current AI/ML experiments employ Python programming, the process module supports Python, Jupyter Notebook, Shell scripting (bash), and SSH for running system-level programs. All the dependent libraries like Deep Learning, Keras, PyTorch, and TensorFlow are easily accessible in the process. The process editor/creator interface allows users to create new processes and edit existing ones. Whenever a new process is created, it gets stored in a MySQL database. The process monitor is responsible for all the execution events in the process module and reports the real-time status. Once the execution is complete the input, output, and code

that has been executed will be recorded and stored in a database. The provenance manager is responsible for evaluating the recorded history of each process in order to assess data quality and recover from failure.

- Workflow: The term "Workflow" is a wide-ranging phrase that can be interpreted in a variety of ways (Jablonski and Bussler, 1996; Van der Aalst, 1998; Kiepuszewski et al., 2003). For instance, many geoscientists often refer to Jupyter Notebook or bash script as a workflow. In Geoweaver, workflow denotes a pipeline linking multiple processes together. The workflow module consists of two functions (1) Building workflows from the existing process and (2) Managing the query, edits, and execution of the workflows. Geoweaver supports not only DAG (Directed Acyclic Graph) workflow, but also other types of workflows such as cyclic, linear, and branching workflows. DAG workflows are commonly used in Geoweaver as they are well-suited for managing complex workflows with many interdependent tasks. However, Geoweaver's support for different workflow types allows users to choose the most appropriate workflow pattern for their specific needs. The workflow module displays a color-coded real-time status of each process in the execution mode. Different colors represent the status of each process: blue means the process is waiting; yellow means the process is running; green means the process is finished running; and red means the process failed. A more detailed demonstration of Geoweaver is described in a previous paper (Kale and Ma, 2023). Exporting and importing the existing workflows in Geoweaver is simple and easy. The downloaded workflow can be automatically loaded into the workspace and ready for execution and reuse. Fig. 1 describes the framework of Geoweaver with the three core modules.

2.3. Architecture of the geoweaver_cwl wrapper tool

There are several workflow management system and languages which are used for expressing workflows into CWL. The Galaxy platform, widely used for managing and analyzing genomic data is a popular system that can translate existing workflows into CWL in order to share and reproduce the existing workflows (Gu et al., 2021). Snakemake is another popular workflow management system in the bioinformatics community where workflows can be automatically exported to CWL (Köster and Rahmann, 2012). Nextflow is a workflow management system designed for big data processing is also used for expressing workflows into CWL (Di et al., 2017). In this paper we designed a wrapper tool named ***geoweaver_cwl***, we regard "wrapper tool" as a process of using CWL to describe command-line tools so that they can be run as an application or a tool in part of a larger workflow. Using the wrapper tool with CWL will make all the documents portable, sharable, and executable. The preliminary step for creating a workflow in Geoweaver is through the workflow module. The workspace allows users to compose a workflow using existing processes. Once the workflow is created it can be downloaded with two options "workflow with process code" or "workflow with process code and history". The first will simply download the workflow and source code. The latter will download all the history of the prior workflow executions in addition to the source code and workflow. The downloaded workflow comes with a Zip file that includes a code folder, a history folder, and a workflow file. The code folder contains the code files (processes) used to form the workflow, the history folder contains the historical details of each process like the begin_time, end_time, input, and output. The workflow file contains the information on the nodes and edges that link together to form the workflow.

To further extend the portability and interoperability of workflows built in the Geoweaver framework, we designed ***geoweaver_cwl***, a Python package that captures inputs (source and target processes) from a Geoweaver workflow file and transforms them into CWL scripts. A key contribution to our work is an add-on functionality that dynamically generates corresponding CWL code without the user having to know the CWL syntax. The CWL file features text fields that comprehensively describe workflow commands and parameters.

Fig. 2 illustrates a brief architecture of ***geoweaver_cwl***. The package contains two main functions "generate_cwl" and "generate_yml". The generate_cwl function takes workflow. json from Geoweaver as the input, captures the nodes, and the edges from the workflow, and writes the steps that form the data flow into CWL scripts. To capture the source and target from the workflow file, we iteratively visit each node in the workflow, and each visited node that has not been previously processed becomes a source node. Then, for each source node, we compile the child nodes, and each child node serves as a target for the source node. Each source-target pair is processed by writing a CWL script that

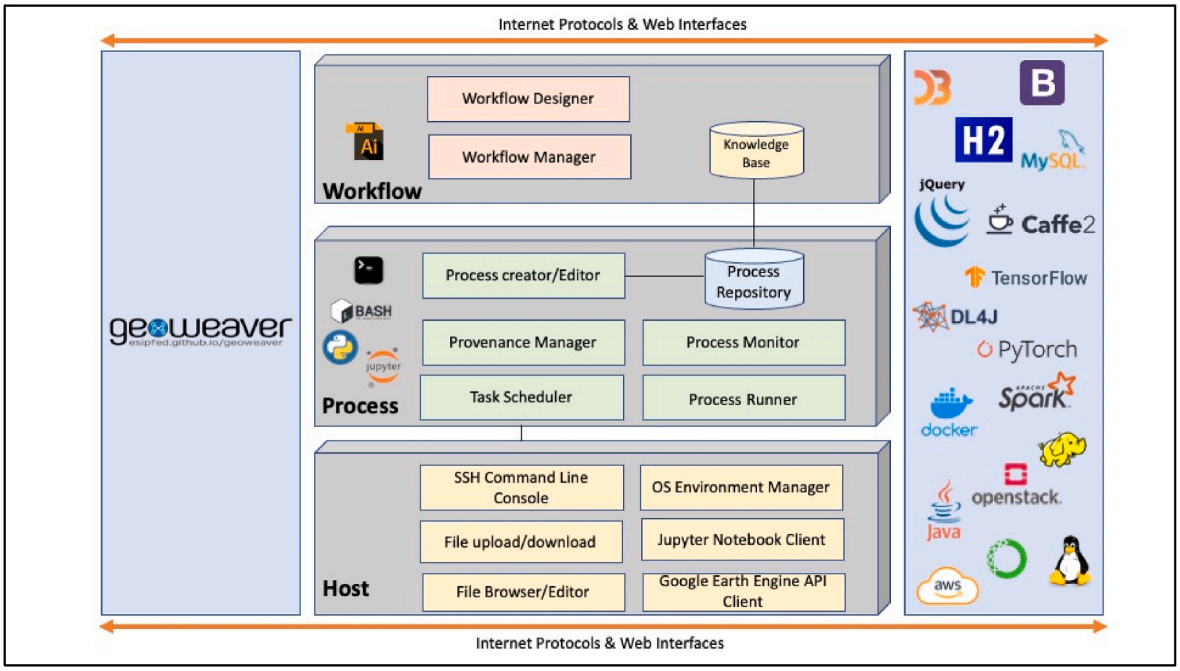


Fig. 1. Workflow management framework of Geoweaver and its core modules (Host, Process, and Workflow), adapted from (Sun et al., 2020).

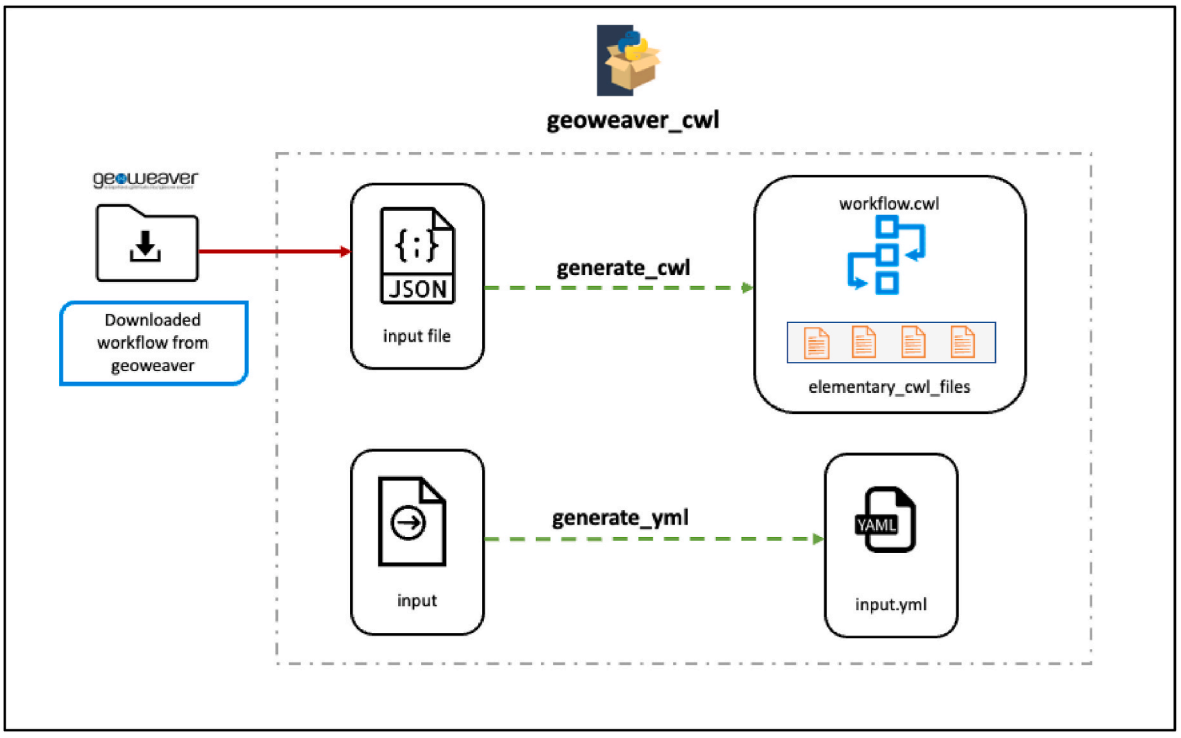


Fig. 2. Architecture of ***geoweaver_cwl*** package with key functions.

provides explicit inputs and outputs for each phase. Carrying out this procedure eventually enables us to generate the CWL scripts for the whole workflow. Equations (1) and (2) below describes the process of translating the workflow file into CWL.

$$PL = \rho(workflow.json) \tag{1}$$

$$[workflow.cwl, elementarycwlfiles] = \forall p : W(p, \in (p)), p \in PL \tag{2}$$

where PL = process_list
ρ = Graph edge extraction function
W, $\in$ = file writing functions

Additionally, the function also generates a new subdirectory called "elementary_cwl_files" which stores new CWL files (the processes used in the workflow) translated from the code folder. Below is the pseudo-code of the generate_cwl and generate_yml functions.

Graph edge extraction function.

```
Read edges from workflow.json
Let process_list, target_list be empty list
      For each edge in edges
      Let source be edge.souce
      Let target be edge.target
            If source not in process_list
                  Append source to process_list
            If source is in target_list
                  Remove source from target_list
            If target not in target_list
                  Append target to target_list
Add elements from target_list to process_list
```

File writing function for workflow. cwl.

```
read process_list -> workflow.json
for process in process_list
      write process_name
      write run command
      write input command
      write output command
      create an elementary_cwl_files
```

File writing function for elementary CWL files.

```
Create a new elementary_cwl_files
      write baseCommand
      write input
      write output
```

The generate_yml function produces a Yet Another Markup Language (YAML) file, which writes the input to run the workflow. cwl file. The YAML file describes which input to run for the cwl files.

```
Class: Directory/file
Path: path of the file or directory
```

The ***geoweaver_cwl*** package is fully open access and the installation is simple. The package can be downloaded from: https://pypi.org/project/geoweaver-cwl/0.0.1/. Fig. 3 demonstrates the installation steps for the ***geoweaver_cwl*** package along with the use of some functions. To facilitate reuse and adaptation, we have made the source code, a detailed user guide, and concrete self-contained examples file available on GitHub under an open-source license: https://github.com/amrutakale08/geoweaver_cwl and self-contained example on https://github.com/amrutakale08/geoweaver_cwl-usecases.

Once the workflow files are described in CWL scripts, they can be

```
In [1]: pip install geoweaver_cwl

        Requirement already satisfied: geoweaver_cwl in /Users/amrutakale/opt/anaconda3/lib/python3.8/site-packages (0.0.9)
        Note: you may need to restart the kernel to use updated packages.

In [2]: from geoweaver_cwl import translator as tr

In [3]: tr.generate_cwl('workflow.json')

        Output file: workflow.cwl
        Writing header...
        Writing steps...
        CWL file written to workflow.cwl

In [4]: tr.generate_yml('input.yml')

        Writing YML file...
        YML file created ...
```

Fig. 3. Installation and usage of the ***geoweaver_cwl*** package.

executed using any other software that supports CWL, like cwltool, Arvados, Toil, CWL-Airflow, and more. In this paper, we are going to use the traditional cwltool. To run the newly generated CWL files from Geoweaver, we will use the below command. We invoke cwl_runner with workflow. cwl and input object input. yml on the command line.

```
cwl-runner workflow.cwl input.yml
```

The command will trigger all the functions inside the CWL and YAML files in the same order as Geoweaver and is supposed to get the same results. As mentioned above, the advantage of CWL is that it provides a solution for describing portable and reusable workflows. The transformation from Geoweaver to CWL through the ***geoweaver_cwl*** package allows geoscientists to easily share, exchange, modify, and reuse workflows. Additionally, CWL-compliant applications are highly portable and can be run in a variety of environments, including local or cloud infrastructures.

3. Use case implementation, result, and evaluation

Based on the ***geoweaver_cwl*** package, we tested a list of workflows from simple to complicated ones. Here we use a Geoweaver workflow available on GitHub (https://github.com/earth-artificial-intelligence/kenya-crop-mask-geoweaver) to demonstrate and verify the usability of our package. The scientific topic of that workflow is the annual and in-season mapping of cropland in Kenya (Tseng et al., 2020). The GitHub repository contains the code folder, history folder, and workflow. json file.

We installed the ***geoweaver_cwl*** package and followed the above-mentioned procedures to describe the workflow in the CWL text document. After using the functions ***generate_cwl*** and ***generate_yml***, we obtained the files "input.yml", "workflow.cwl", and "elementary cwl files folder", which included the cwl files used in creating the workflow. The workflow translation process was fast and easy, and we also noticed that using cwltool speeds up workflow execution compared to the original procedure in Geoweaver. Yet, we still need to run more use cases to see

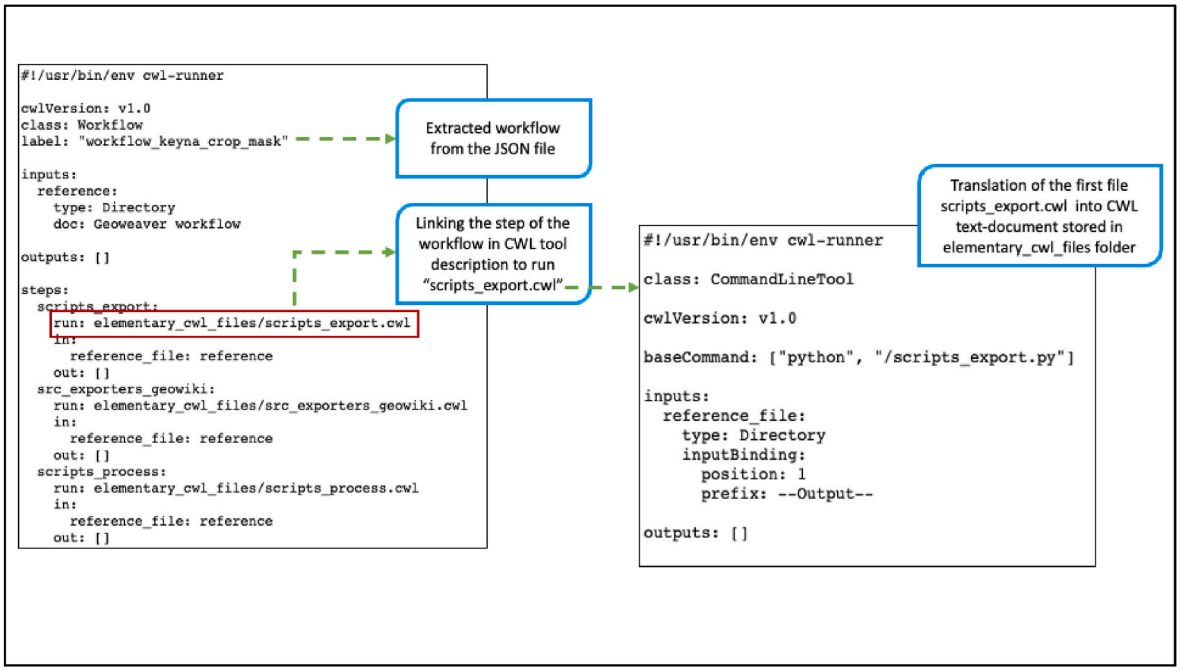


Fig. 4. Exemplar scripts of workflow steps in the workflow. cwl from workflow. json file (left) and the CWL text document scripts_export.cwl describing computational steps present in the elementary_cwl_files folder (right).

whether CWL and cwltools always have shorter execution time comparing with Geoweaver.

We successfully transformed the Geoweaver workflow of Kenya cropland mapping into CWL format using the ***geoweaver_cwl*** package. The left part of Fig. 4 shows the described workflow in CWL from the workflow. json in Geoweaver. The CWL file contains a ***cwlVersion*** section which indicates the version of the CWL document. The ***class*** section with a value of ***Workflow*** indicates that this document describes the workflow. The ***inputs*** and ***outputs*** sections describe the inputs and outputs of the workflow, respectively. The ***steps*** section describes the actual steps of the workflow. In this example, the first step is to run the "scripts_exports.cwl" present in the folder elementary_cwl_files. The code of "scripts_exports.cwl" is illustrated in the right part of Fig. 4. The workflow steps in CWL do not always run in the written sequence. Instead, the order is determined by the dependencies across steps. To evaluate the result of the transformation we ran the CWL text document using cwltool, and we observed that it executed smoothly and generated the same result as in Geoweaver. The CWL result of this example is accessible on GitHub: https://github.com/amrutakale08/geoweaver_cwl-usecases. We are now transforming more Geoweaver AI workflows into CWL with this package and sharing the results on GitHub. Interested readers can go to that GitHub repository through the above link to test and adapt those use cases.

Geoweaver provides a unique combination of features, such as a user-friendly interface, full-stack code, a history of previous versions, and sharable AI/ML workflows. It is a user-friendly entry point to solve AI-related workflow issues for a variety of disciplines in geosciences as well as beyond. The ***geoweaver_cwl*** package developed in this work further extends the portability and interoperability of workflows created in Geoweaver. The package can quickly transform Geoweaver workflows into CWL format, and the result can be run on many CWL-compliant software applications. Moreover, the CWL result can be also executed on diverse computing platforms including local computers, cloud environments, or high-performance clusters. The transformation process is intuitive and new users will spend less time getting familiar with the package.

4. Discussion

We encourage geoscientists as well as other AI practitioners to use Geoweaver and the ***geoweaver_cwl*** package to increase the reproducibility and interoperability of their work. The developed package helps automatically transform Geoweaver AI/ML workflows to a community standard CWL. As an extension to Geoweaver, the CWL result can be executed on diverse computing platforms which gives users more opportunities to run the workflow without compromising provenance or having to recreate the workflow if they want to use another workflow management system. CWL can formally describe inputs, outputs, and other execution details of the workflow in a text-based document. It supports workflows that specify dependencies among tools and use one device output as input to another. CWL documents are text-based so that they can be created manually, without or with less computer programming. However, ensuring that these documents adhere to the CWL syntax specification may restrict some users from adopting it. The developed ***geoweaver_cwl*** addresses this gap. It can automatically describe workflows into CWL to make it effortless for geoscientists to share data analysis workflows in varied formats without learning the technical details of the CWL syntax.

There are a wide variety of workflow management system software tools available all over the research community, that are constantly being developed, revised, and improved every day. While the availability of such tools benefits the community, it also presents a great challenge: as more and more tools are created, a set of standards needs to be adopted in order to ensure the portability and reproducibility of the resulting workflows. CWL, as reflected in its name, aims to be such a community standard to harmonize the workflow formats proposed by various workflow management system software tools. Reproducibility enables researchers to track and debug potential errors and validate the authenticity of the results, and as such it plays a vital role to make scientific research accurate, efficient, and cost-effective. Because CWL tracks code versions, inputs, outputs, and more, researchers can use it to pinpoint where the analysis went wrong, or where in the analysis the particular piece of data leads to new insights. Therefore, the transformation from Geoweaver workflows to CWL format is a necessary extension with regards to broad portability and reproducibility.

Portability is crucial when it comes to scientific research and analysis. When one workflow is designed for a type of computational environment such as a personal computer it may not function in a similar way as in the cloud. Therefore, researchers may spend more time and effort in debugging the tool to make it work in the desired environment. This could result in inconsistent outcomes or errors. In contrast, CWL enables portability by being explicit about inputs, outputs, data location, and execution models that can be executed on any of the CWL-compliant environments. CWL-based documents can be downloaded, edited, and executed on local infrastructure or uploaded and executed in the cloud.

The scientific provenance research community has evolved significantly in recent years to provide several strategic capabilities, to make AI/ML workflows more explainable and reproducible. The declarative approach to describe workflow in CWL scripts facilitates and encourages users to explicitly declare every single step, improving the white box view of reviewing process and potential provenance. Such workflows will eliminate the "black box" nature by offering insights into the entire process used to build artifacts. This will support the research community in carrying out thorough studies that will enable them to satisfy those essential requirements for building a transparent and explainable AI/ML application. Documenting provenance to support published research should be considered a best practice rather than an afterthought. The community should be encouraged to follow well-established and consensus best practices for workflow design and software environment deployment. The aim of Geoweaver and the ***geoweaver_cwl*** package is to promote the efforts in that direction.

In order to improve the efficiency of the developed ***geoweaver_cwl*** package, our plan is to continue using Geoweaver and the package with more AI research projects. So far, we have only tested our package on definite workflows created by Geoweaver, and we believe further analyses are necessary to validate the broad utility of the package. For instance, with the small number of use cases of ***geoweaver_cwl*** application we noticed that the CWL and cwltools have shorter execution time comparing with Geoweaver. Nevertheless, the diverse datasets, algorithms, and workflow may lead to varied performance, so we need to do more tests to see if that shorter execution time is always true. For our future work, we would like to collaborate with a diverse research team from different domains and collect complex use cases from them. Testing different use cases will confirm additional details and novel functions and also ensure that our package satisfies the end-user requirement. Geoweaver is developed and implemented using Java. A plan under discussion among our team is to have a new version of Geoweaver in Python, called "pygeoweaver". In that way the Python-based ***geoweaver_cwl*** package can be naturally included as part of the new "pygeoweaver" platform, to address the needs from both Geoweaver and the CWL communities.

5. Conclusions

In this paper, we first introduced Geoweaver and presented a wrapper tool, called ***geoweaver_cwl***, that overcomes current challenges of achieving repeatability, reproducibility, and reusability of workflows. To assess the outcome, we tested ***geoweaver_cwl*** with multiple use cases provided by Geoweaver and illustrated one of them in this paper. The study demonstrates that the ***geoweaver_cwl*** package can bring great benefits to the geoscience community. The code is publicly available on GitHub (https://github.com/amrutakale08/geoweaver_cwl) and open

to anyone who wants to import Geoweaver AI/ML workflows into CWL-compliant workflow management system software applications. We encourage the research community to participate in the adoption of Geoweaver by integrating the ***geoweaver_cwl*** package into their projects. We would like to hear comments and suggestions from the community to facilitate the development of new functionality in future versions.

Code availability

The geoweaver_cwl Python package is made open access at: https://pypi.org/project/geoweaver-cwl/0.0.1/. The source code of the package is accessible at: https://github.com/amrutakale08/geoweaver_cwl and exemplar results are accessible at: https://github.com/amrutakale08/geoweaver_cwl-usecases. The source code of the Geoweaver platform is accessible at: https://github.com/ESIPFed/Geoweaver.

CRediT authorship contribution statement

Amruta Kale: Writing – review & editing, Writing – original draft, Software, Methodology, Conceptualization. **Ziheng Sun:** Writing – review & editing, Resources, Methodology. **Chao Fan:** Writing – review & editing, Resources, Methodology. **Xiaogang Ma:** Writing – review & editing, Validation, Methodology, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data and code are shared on GitHub and links were provided in the manuscript.

Acknowledgment

The work was supported by the National Science Foundation under Grants No. 2019609 and No. 2126315 and the National Aeronautics and Space Administration under Grant No. 80NSSC21M0028. The authors thank two anonymous reviewers for their reviews and comments on an early version of the paper.

References

Adadi, A., Berrada, M., 2018. Peeking inside the black box: a survey on explainable artificial intelligence (XAI). IEEE Access 6, 52138–52160.

Altintas, I., Berkley, C., Jaeger, E., Jones, M., Ludascher, B., Mock, S., 2004. Kepler: an extensible system for design and execution of scientific workflows. June. In: Proceedings of the 16th International Conference on Scientific and Statistical Database Management. Santorini, Greece, pp. 423–424.

Amstutz, P., Crusoe, M.R., Tijanić, N., Chapman, B., Chilton, J., Heuer, M., Kartashov, A., Leehr, D., Ménager, H., Nedeljkovich, M., Scales, M., 2016. Common Workflow Language, v1.0. Figshare. https://doi.org/10.6084/m9.figshare.3115156.v2.

Bedia, J., San-Martín, D., Iturbide, M., Herrera, S., Manzanas, R., Gutiérrez, J.M., 2019. The METACLIP semantic provenance framework for climate products. Environ. Model. Software 119, 445–457.

Das, P., Ivkin, N., Bansal, T., Rouesnel, L., Gautier, P., Karnin, Z., Dirac, L., Ramakrishnan, L., Perunicic, A., Shcherbatyi, I., Wu, W., 2020. Amazon SageMaker Autopilot: a white box AutoML solution at scale. In: Proceedings of the Fourth International Workshop on Data Management for End-To-End Machine Learning, pp. 1–7. Portland, OR,USA.

DataRobot, AI Cloud, 2012. https://www.datarobot.com/. (Accessed 18 January 2022).

Di Tommaso, P., Chatzou, M., Floden, E.W., Barja, P.P., Palumbo, E., Notredame, C., 2017. Nextflow enables reproducible computational workflows. Nat. Biotechnol. 35 (4), 316–319.

Doshi-Velez, F., Kortz, M., Budish, R., Bavitz, C., Gershman, S., O'Brien, D., Scott, K., Schieber, S., Waldo, J., Weinberger, D., Weller, A., 2017. Accountability of AI under the law: the role of explanation. arXiv preprint arXiv:1711.01134.

Gilpin, L.H., Bau, D., Yuan, B.Z., Bajwa, A., Specter, M., Kagal, L., 2018. Explaining explanations: an overview of interpretability of machine learning. In: Proceedings of the 2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA). Turin, Italy, pp. 80–89.

Gu, Q., Kumar, A., Bray, S., Creason, A., Khanteymoori, A., Jalili, V., Grüning, B., Goecks, J., 2021. Galaxy-ML: an accessible, reproducible, and scalable machine learning toolkit for biomedicine. PLoS Comput. Biol. 17 (6), e1009014.

Gunning, D., Aha, D., 2019. DARPA's explainable artificial intelligence (XAI) program. AI Mag. 40 (2), 44–58.

Jablonski, S., Bussler, C., 1996. Workflow Management: Modeling, Concepts, Architecture and Implementation. Cengage Learning.

Kale, A., Ma, X., 2023. Provenance in earth AI. In: Sun, Z., Cristea, N., Rivas, P. (Eds.), Artificial Intelligence in Earth Science. Elsevier, pp. 357–378. Amsterdam.

Kale, A., Nguyen, T., Harris Jr., F., Li, C., Zhang, J., Ma, X., 2023. Provenance documentation to enable explainable and trustworthy AI: a literature review. Data Intelligence 5 (1), 139–162. https://doi.org/10.1162/dint_a_00119.

Kiepuszewski, B., Barros, A.P., Van Der Aalst, W., Ter Hofstede, A., 2003. Workflow patterns. Distributed Parallel Databases 14 (1), 5–51.

Köster, J., Rahmann, S., 2012. Snakemake—a scalable bioinformatics workflow engine. Bioinformatics 28 (19), 2520–2522.

Ma, X., Beaulieu, S.E., Fu, L., Fox, P., Di Stefano, M., West, P., 2017. Documenting provenance for reproducible marine ecosystem assessment in open science. In: Diviacco, P., Glaves, H.M., Leadbetter, A. (Eds.), Oceanographic and Marine Cross-Domain Data Management for Sustainable Development. IGI Global, Hershey, PA, USA, pp. 100–126.

Datatron MLOps, 2016. Machine learning operations. https://datatron.com/. (Accessed 18 January 2022).

Pahl, C., Brogi, A., Soldani, J., Jamshidi, P., 2017. Cloud container technologies: a state-of-the-art review. IEEE Transactions on Cloud Computing 7 (3), 677–692.

Ribeiro, M.T., Singh, S., Guestrin, C., 2016. Why should I trust you? Explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. San Francisco, CA, USA, pp. 1135–1144.

Rudin, C., 2018. Please stop explaining black box models for high stakes decisions. In: Proceedings of the 32nd Conference of Neural Information Processing Systems (NIPS), Workshop on Critiquing and Correcting Trends Machine Learning. Montreal, Canada, 20pp. Available: https://arxiv.org/abs/1811.10154.

Rudin, C., 2019. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nat. Mach. Intell. 1 (5), 206–215.

Sun, Z., Di, L., Burgess, A., Tullis, J.A., Magill, A.B., 2020. Geoweaver: advanced cyberinfrastructure for managing hybrid geoscientific AI workflows. ISPRS Int. J. Geo-Inf. 9 (2), 119. https://doi.org/10.3390/ijgi9020119.

Sun, Z., Sandoval, L., Crystal-Ornelas, R., Mousavi, S.M., Wang, J., Lin, C., Cristea, N., Tong, D., Carande, W.H., Ma, X., Rao, Y., Bednar, J.A., Tan, A., Wang, J., Purushotham, S., Gill, T.E., Chastang, J., Howard, D., Holt, B., Gangodagamage, C., Zhao, P., Rivas, P., Chester, Z., Orduz, J., John, A., 2022. A review of earth artificial intelligence. Comput. Geosci. 159, 105034 https://doi.org/10.1016/j.cageo.2022.105034.

Tseng, G., Kerner, H., Nakalembe, C., Becker-Reshef, I., 2020. Annual and in-season mapping of cropland at field scale with sparse labels. Tackling climate change with machine learning workshop at NeurIPS '20, virtual conference. https://www.climatechange.ai/papers/neurips2020/29. (Accessed 17 November 2022).

Van der Aalst, W.M., 1998. The application of Petri nets to workflow management. J. Circ. Syst. Comput. 8 (1), 21–66.

Wing, J.M., 2020. Ten research challenge areas in data science. Harvard Data Science Review 2 (3). https://doi.org/10.1162/99608f92.c6577b1f.

Zeng, Y., Su, Z., Barmpadimos, I., Perrels, A., Poli, P., Boersma, K.f., Frey, A., Ma, X., de Bruin, K., Gossen, H., Timmermans, W., 2019. Towards a traceable climate service: assessment of quality and usability of essential climate variables. Rem. Sens. 11 (10), 1186. https://doi.org/10.3390/rs11101186.