Towards GeoAI as a Containerized Microservice

Jiapan Wang

Professorship of Big Geospatial Data Management, Technical University of Munich
Munich, Germany
jiapan.wang@tum.de

ABSTRACT

Geospatial Artificial Intelligence as a containerized microservice (GeoAIaaS) utilized microservice-based architecture for GeoAI applications by pre-defined mission recipes of handling geospatial data to lower the additional geography expertise barrier and improve the reusability of GeoAI methods. This paper conducts standalone, distributed developable, and simply deployable microservices for three phases of geospatial object detection application. The potential of GeoAIaaS across local workstations, edge computing, and distributed development is investigated and particularly the microservice for processing remote sensing images and OSM data is found to improve the capability of the open data sources. The code is available via https://github.com/Wjppppp/osm-building-detection-server.git.

CCS CONCEPTS

• Computing methodologies → Distributed artificial intelligence.

KEYWORDS

GeoAI, deep learning, microservices, object detection, reusability

ACM Reference Format:


1 INTRODUCTION

GeoAI has recently emerged as a research hotspot among researchers working with data mining, machine learning, and high performance computing in the field of big geospatial data analysis. With the rising availability of satellite imagery and advancement of deep learning (DL) techniques, demand for GeoAI applications is growing not only in the research domain but also in practical production, such as classification, object detection, semantic segmentation, etc [4].

A major challenge with applying GeoAI today is the reusability of methods for geospatial data preparation and processing. For instance, while applying deep learning methodologies to remote sensing data, which often contains additional spectral bands beyond RGB and potentially integrates with other geospatial data that may have diverse coordinate systems, the complexity of handling various geospatial data should not be underestimated. Large companies have realized this general problem in the AI field and offer easy-to-use Application Programming Interfaces (APIs) [5], such as Hugging Face, JINA AI, etc. Recent research in formulating the application of GeoAI into high-level programming libraries has risen to lower the barrier of geography-related knowledge and programming efforts across various deep learning frameworks, such as TorchGeo [6], etc. However, these libraries have four significant limitations: (a) Tailoring them for specific applications isn’t straightforward. (b) They are usually cloud-based. (c) They are complex to understand and use due to the lack of tutorial workf lows or incomplete documentation. (d) They lack support for reusability and configuration sharing across different applications.

To address these challenges, this paper introduces GeoAI as a containerized microservice (GeoAIaaS) that is individually developable, locally runnable, reusable, and simply deployable. The main idea is to pack up frequently used GeoAI methods for geospatial data collection, data processing, and model prediction through well-defined interfaces. So that users can have access to the GeoAI workflow efficiently from anywhere. The proposed GeoAIaaS framework primarily targets small research institutes or individual researchers, who typically need to learn new technologies when building their ML solutions. Through the employment of pre-defined microservices, users can focus more on the data itself without being concerned about the implementation across diverse programming libraries. Leveraging the reproducible capability of Docker [1], the developed GeoAI microservice can be effortlessly deployed on local workstations, cloud-based infrastructures, or edge computing platforms.

2 ARCHITECTURAL DESIGN

A GeoAI application typically comprises three parts: frontend, backend, and microservices. We depict an easy-to-build architectural framework of a microservice-based GeoAI application and APIs between different parts (Figure 1).

**Frontend** - As the main entrance of the GeoAI application, the frontend should adhere to a user-centered design, aiming at providing a user-friendly and intuitive portal to let users interact with maps. Several tools like Leaflet, and Cesium enable the integration of OGC Web Map Service (WMS) provided by various map providers like Bing Maps, OpenStreetMap (OSM), and Google Maps.

**Backend** - Responsible for handling server-side functionalities that support data management, communication, and API exposure. In our case, it provides REST APIs that enable the communication of geospatial data, model configuration, and mission recipes between multiple microservices.

**Microservice** - It plays a significant role in GeoAI applications by providing modular, scalable, reusable, and distributed developable geospatial data analysis services. Each microservice can be
tailored to handle specific missions via pre-defined recipes, such as processing remote sensing imagery, querying external geographic data, preparing training datasets, conducting object detection, performing semantic segmentation, etc. The key feature of GeoAI as a microservice is that it can hide the geographic-related process inside itself to lower the knowledge barrier for researchers. Additionally, it streamlines the intricacies inherent in programming workflows for processing big geospatial data. For example, if someone develops a microservice that splits remote sensing images into fix-sized image tiles with location embedding, given proper metadata, this microservice can be universally employed across various GeoAI applications requiring remote sensing image tiles.

3 CASE STUDY

Geospatial object detection, a prominent research area within GeoAI, has gained substantial development due to the growing availability of remote sensing imagery and advancements in AI techniques, showcasing the great potential for practical applications [2]. Our paper conducts the full workflow of utilizing GeoAI methods to solve building detection from satellite imagery in three phases: data preparation, training, and prediction (Figure 1). Each phase is wrapped into a standalone microservice, which can be called via a REST API from the backend.

**Data Preparation** - This microservice can query and split remote sensing images sourced from Bing Imagery Service or other WMS providers into the desired size without losing location information and if needed, query geometries from OpenStreetMap [3]. Furthermore, it can wrap image and geometry data into a specific data structure, like AtlasHDF [7] or other ready-for-training data.

**Training** - This microservice bridges datasets and DL models, providing the development environment and computation resource for training DL models. It can query a pre-trained object detection model from open sources, like Tensorflow Object Detection Model Zoo, and train it with prepared training records composed of remote sensing images and OSM geometries. The training pipeline can be tailored within the microservice and executed via HTTP request.

**Inferencing** - To efficiently deploy GeoAI applications, this microservice provides the interface for predicting buildings by trained models within the given geographic coordinates. The predicted bounding boxes and confidential scores will be returned in the easy-to-transfer geospatial data structure, like GeoJSON, etc.

4 CONCLUSION

GeoAI as a containerized microservice was shown to be an efficient and easy-to-maintain framework for both GeoAI research and applications. It splits the whole workflow of GeoAI into three phases conducted by diverse microservices. Each microservice is standalone and simply deployable, aiming at lowering the geography knowledge barrier and improving the reusability of GeoAI applications. The experiment of geospatial object detection application shows the potential of building self-served spatial datasets, efficiently training processes, and locally runnable GeoAI services. However, since the complexity and high data volume of big geospatial data, the toleration of pre-defined microservices is still needed when facing different problems. Future work will explore the standard of building a GeoAI microservice and leverage the community to support more use cases. The findings of this paper offer insights into establishing a collaborative GeoAI workflow for small research groups or independent developers and shed inspiring light on possibilities for GeoAI applications across local workstations, edge computing, and distributed development.

REFERENCES


Figure 1: The architecture of GeoAI as a Microservice and a use case of geospatial object detection web application.