INL/CON-23-72297-Revision-0

Supporting Nuclear Energy Research with MLOps

October 2023

Brandon Samuel Biggs Jr, Hermann, Dunya Bahar

Changing the World's Energy Future

INL is a U.S. Department of Energy National Laboratory operated by Battelle Energy Alliance, LLC

INL/CON-23-72297-Revision-0

Supporting Nuclear Energy Research with MLOps

Brandon Samuel Biggs Jr, Hermann, Dunya Bahar

October 2023

Idaho National Laboratory
Idaho Falls, Idaho 83415

http://www.inl.gov

Prepared for the
U.S. Department of Energy
Under DOE Idaho Operations Office
Contract DE-AC07-05ID14517

Supporting Nuclear Energy Research with MLOps

1st Brandon Biggs
Advanced Scientific Computing
Idaho National Laboratory
Idaho Falls, United States
brandon.biggs@inl.gov

2nd Dunya Bahar
Advanced Scientific Computing
Idaho National Laboratory &
Idaho State University
Idaho Falls, United States
dunyahermann@isu.edu

***Abstract*—Artificial intelligence (AI) and machine learning (ML) have been shown to be increasingly helpful tools in a growing number of use-cases relevant to scientific research, despite significant software-related obstacles. There exist large technical costs to setting up, using, and maintaining AI/ML models in production. This often prevents researchers from utilizing these models in their work. The growing field of machine learning operations (MLOps) aims to automate much of the AI/ML life cycle while increasing access to these models. This paper presents the initial work in creating a nuclear energy MLOps platform for use by researchers at Idaho National Laboratory (INL) and aims to reduce the barriers of using AI/ML in scientific research. Our goal is to promote the integration of the latest AI/ML technologies into researchers' workflows and create more opportunity for scientific innovation. In this paper we discuss how our MLOps efforts aim to increase usage and the impact of AI/ML models created by researchers. We also present several use-cases that are currently integrated. Finally, we evaluate the maturity of our project as well as our plans for future functionality.**

***Index Terms*—Machine learning, artificial intelligence, machine learning life cycle, machine learning operations, DevOps, MLOps, developer productivity, nuclear energy, FastAPI, transformers, PyTorch**

I. Introduction

For many tasks, AI/ML techniques are a huge leap forward from traditional methods. In general, AI/ML uses large amounts of data to find patterns, detect anomalies, make predictions, and classify new data. Since many models are able to learn from data, they can be updated to adapt to new information as real-world data changes. Unlike statistical or rule-based models, the results generated from neural network based ML models do not require long lists of rules to be manually updated. Instead, these systems can deduce complex mathematical relationships from elaborate high-dimensional data. These models are typically difficult to explain (often referred to as "black boxes"), and are not usually easily inferred via traditional statistics. Researchers could greatly benefit from wider access to these models, but most AI/ML models are not widely usable due to the persisting roadblocks to deployment.

Innovations in artificial intelligence, natural language processing (NLP), computer vision, and phsyics-informed neural networks [22] have expanded the use-cases for AI/ML in nuclear energy and safety [23]. Integrating AI/ML into scientific workflows has shown exciting potential to aid software-enabled discovery [24]. Some software solutions attempt to bridge the technological gap between models and users, but these tools are often not user-friendly, require complex dependencies, or are not well suited for a research context. Thus, integration of AI/ML into scientific workflows requires considerable development of software.

Without an MLOps system in place, organizations which hope to utilize AI/ML would typically need to budget for months of up-front work just to establish a functional model in addition to long-term maintenance costs [4]. This happens because the code to make an existing ML model usable is, surprisingly, often larger and more complex than the models themselves. Deploying even one model can take companies between 8 and 90 days [4]. Additionally, deployment attempts often fail altogether due to lack of expertise and high costs [4]. These roadblocks are unfortunate because there are far-reaching potential benefits of AI/ML which could reduce research time and open new avenues to scientific discovery [7].

Researchers at INL hoping to utilize AI/ML in their scientific workflows are currently limited in their options. Domain experts may not have the resources or hardware needed to host a reliable AI/ML model for their specific use-case. Acquiring funding to hire developers to establish a production system is also challenging. The software infrastructure required to get a model into production is often not justifiable or practical for an individual project. Also, there are not robust inference tools online that can stand in for a custom-built MLOps pipeline. Data sensitivity, privacy, and regulations [25] in scientific research are additional hurdles that need to be overcome. These roadblocks to utilization clearly indicated a need for a more widely-accessible platform at INL which would unlock the benefits of AI/ML for use. This work presents initial efforts toward building an MLOps platform which will significantly lower the technical burden of utilizing existing AI/ML models in scientific research.

We have defined five primary goals and outcomes of an MLOps platform for the benefit of nuclear energy researchers and collaborators at INL:

1) Lower the barrier to entry for using AI/ML in everyday research work
2) Simplify the production process for AI/ML engineers and data scientists
3) Enable collaboration and visibility via a central platform for uploading and sharing models

4) Reduce repeated work by eliminating the need to recreate models from scratch in order to reproduce results
5) Increase impact, longevity, and reach of models created at INL through an accessible model hosting platform

This work is structured as follows: in Section II we provide a brief background about INL, and discuss impacts the MLOps platform will have on nuclear energy research. We explore the issue of low adoption of AI/ML in real-world applications, and investigate a root cause of this issue. In Section III we describe the architecture and design choices of our proposed MLOps system. In Section IV we explore relevant use-cases of an MLOps platform. Lastly, in Section V we discuss MLOps maturity, provide suggestions for future work, and conclude.

II. Background

Idaho National Laboratory employs over 5500 people, many of which are working on nuclear energy related research. The high performance computing (HPC) organization at INL supports approximately 900 users on four HPC systems. These users come from national laboratories, universities, and industry partners. The HPC systems were utilized for over 800 million core hours supporting nuclear energy research in fiscal year 22 [1]. Additionally, these systems were used to perform 13.6 million core hours on hardware that contains AI/ML accelerators. Furthermore, AI/ML related work has continued to grow at INL. The increased growth and interest in graphics processing unit (GPU) accelerated workloads is another motivation for creating the MLOps platform to support research.

There are numerous use-cases of AI/ML in nuclear energy research. AI/ML can be used to model complex dynamics of nuclear materials in various conditions [9], maintain safety at nuclear power plants [10]–[12], segment defects in transmission electron micrographs [2], and model nuclear reactors and powerplants through the use of digital twins [27]. Surveys have compiled many real-world applications of AI/ML for nuclear energy research which show the many nuclear applications that would benefit from AI/ML methods, but many of these require an understanding of advanced AI/ML techniques and nuclear-specific industry knowledge [15], [16] . Our MLOps platform will enable wider adoption of these applications at INL to support the work of researchers.

MLOps includes many similar principals to that of development operations (DevOps) [19], including efficient development of software and continuous integration and development (CI/CD). MLOps diverges from DevOps by also including the unique aspects of AI/ML software development. For example, versioning of models and datasets are complex due to large amounts of data and changes that occur during the iterative process of data preprocessing, hyperparameter tuning, and dimensionality reduction. Additionally, MLOps promotes the incorporation of auto-monitoring for model drift which occurs when data the model was originally trained on no longer reflects real-world conditions. MLOps practices would monitor for a decreased performance of the model, and may trigger automatic re-training on up-to-date data or simply alert maintainers of model degradation.

The prohibitive time-costs of moving models into production causes some adoption issues within AI/ML research. Utilization is low due to roadblocks like inaccessible models, inconsistent formatting of data, poor documentation around implementation details, and difficulty in learning relevant tools [7]. To make models accessible at scale such that others can easily iterate on new methods, many different technologies are required. These technologies involve working knowledge of databases, security and authentication, command line scripting, CI/CD, unit and load testing, front-end web frameworks, HTTP web requests, memory and storage management, software architecture and design best practices, and containerized workload tools like Docker [18] or Kubernetes [17]. The difficulty of taking a model from a raw form on one researcher's system and making it available for others to use contributes to the slow real-world adoption of AI/ML.

Furthermore, difficulty of serving a machine learning model negatively impacts scientific innovation. Many research publications describing software will include a link to a web application or code repository that interested parties can view or utilize. Unfortunately, some AI/ML research has further challenges with utilization. AI/ML researchers may be unable to provide access to data or model weights due to privacy concerns or classification, and even if the data and weights are available, the hardware requirements to run these models may not be accessible. This makes it difficult to test, utilize, or apply new methods described in publications [7]. There does not yet seem to be an elegant solution to this issue, so researchers hoping to validate or build off the work must spend extra time attempting to recreate those results from scratch. This repeated work could be prevented via a more accessible means of hosting and sharing models.

These obstacles are slowing the rate of AI/ML innovation and challenging the reproducibility of nuclear energy research, so the need for an MLOps platform has grown. AI/ML engineers and data scientists have needed to quickly iterate on the findings of other research without repeating work and in a reproducible manner to that in the original research. This platform provides a central location for researchers to share AI/ML models, with the additional benefit of increasing the lifespan and impact of models. The MLOps platform aims to make it easier to create and keep a model usable for the benefit of others, after the original creators may have moved on to other work.

III. Architecture

Many open source tools were evaluated and several tools that best fit the project goals were selected. These tools have been used together to create an initial implementation of an MLOps platform to support nuclear energy research. The availability of these tools has enabled a faster development process and allowed us to build our own system as we needed a simple and consistent deployment pipeline, the option for online or batch inference, and a central hub for models.

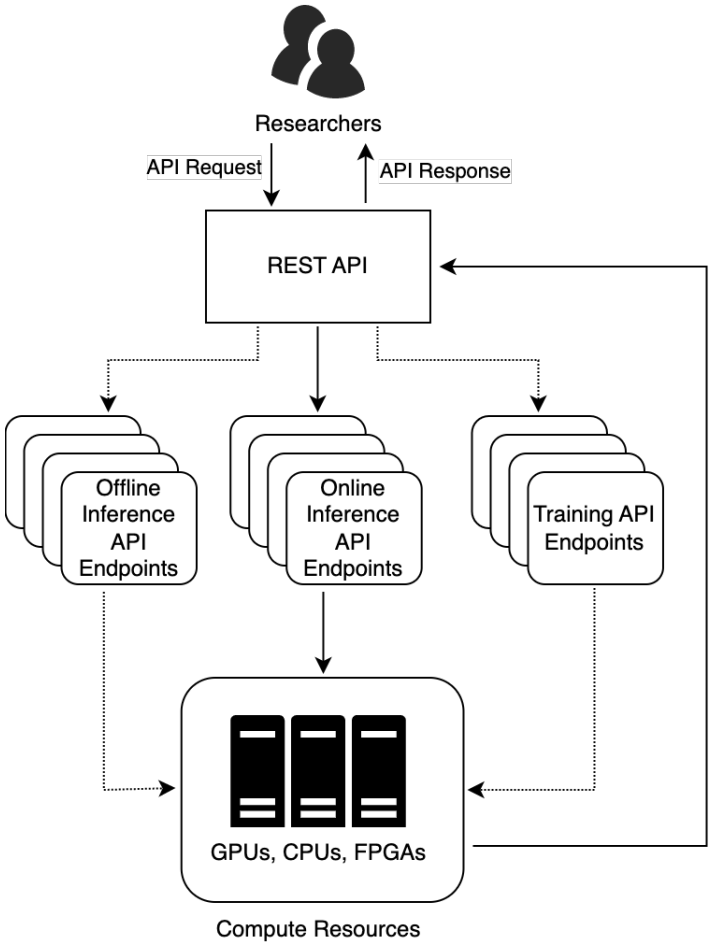

Fig. 1. The proposed design of a centralized API that connects to individual model endpoints which can run on various hardware resources.

Researchers can use the models at their most recent and best performing training iteration, with all the ease of use, security, and fine-tuning that they expect in order to do their research well. To best describe the creation and implementation of an MLOps platform for use in nuclear energy research at INL, we outline the industry tools used in development, describe the system's architecture, and outline our process for system design decisions. By combining these separate tools together to create a unified platform, researchers can experience a cohesive AI/ML workflow. In the following subsections, the architecture of the current and proposed MLOps platform is outlined and a brief discussion of applied industry tools is provided.

A. System Design

Figure 1 shows the proposed design of a centralized application programming interface (API) that connects to individual model endpoints which can run on various hardware resources. The offline inference endpoints would be used to run many inference tasks when the resources are available, such as over the weekend. The online inference endpoints are utilized for single inference tasks, and the training endpoints would be for data ingestion and retraining of models. The solid lines specify current functionality while the dotted lines indicate future additions. Because of the various hardware requirements for training and inference, each model has its own endpoint that can run on the appropriate hardware. Each model endpoint also exists in an isolated container to reduce the burden of model dependencies. Additionally, the model endpoint containers can be scaled to run on many hosts if necessary. The private inference endpoints are called from the central public API. Each model endpoint communicates with a central API that monitors each model and provides researcher access. A user would then make inference requests via a representational state transfer (REST) API. Currently only online inference pipelines have been developed. However, functionality for offline inference and training is planned. Scenarios where a researcher needs a larger batch of work done without an urgent deadline would benefit from offline inference. Additional training endpoints will be used for model retraining in the situation that model drift is detected or new data is added and model improvements can be made. While the current endpoints use various compute resources such as GPUs and CPUs, the endpoints are not hardware restricted.

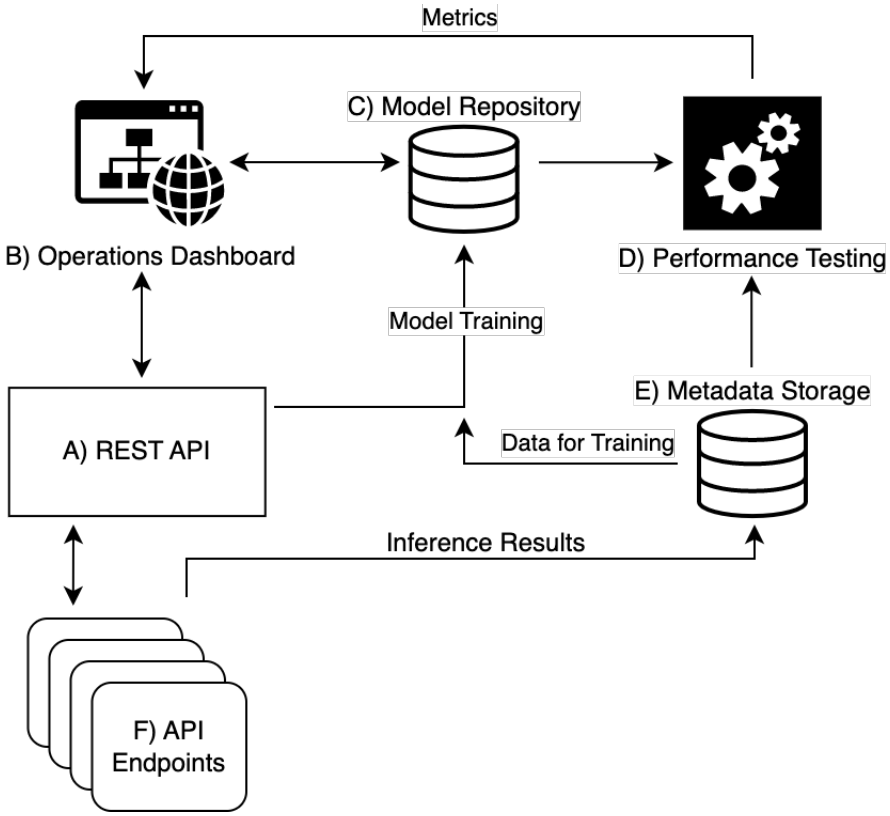

Fig. 2. The proposed back-end architecture of the MLOps platform.

More specialized hardware such as tensor processing units (TPUs) [20] could be used as well as other AI accelerating hardware [21]. Plans to allow field programmable gate arrays (FPGAs), which can run in low-power environments, are also being developed.

Figure 2 shows the additional architecture needed to handle the model life cycle. Label A shows the primary REST API interface that is first called when inference or training is desired. The operations dashboard at label B handles general create, read, update, and delete (CRUD) operations and monitoring, CI/CD pipelines, and API logging. Label C shows the model repository, where models are stored. Label D represents a system for performance testing, which is where the precision and accuracy of the inference results are checked. Label E provides an instance of a storage system where metadata can be stored. Other aspects of the machine learning life cycle can benefit from additional metadata, such as the retraining process. Label F shows the API endpoints in a minimized display of Figure 1.

B. Industry Tools

Of the available industry tools available for implementing MLOps, one primary API framework was selected for inference endpoints. Our choices were informed by a need for flexibility to interface with models that were written in languages other than Python as well as models that used industry standard AI/ML frameworks. FastAPI [13] was used to build a REST API that interfaces with inference endpoints that were also built with this framework. FastAPI is a widely-used modern Python framework for building APIs. Some benefits include automatically generated documentation via OpenAPI standards, high performance, and the utilization of standard Python type hints. Development with FastAPI is fast and straightforward, thus the selection of this framework was also influenced based on time to an initial solution. In addition to FastAPI, many standard AI/ML Python frameworks were used to create and load models, such as Tensorflow, Pytorch, and Hugging Face's transformers package. These choices were heavily influenced by the experiences that the INL team had. While these tools represent a small portion of available open source tools, the MLOps platform was designed to be extensible, meaning that any other AI framework or even non Python application could be added without a redesign or significant changes of the system.

To demonstrate the simplicity of using FastAPI, a code snippet in Figure 3 has been included. That code is the entirety of a real endpoint for classifying an image using transformers.

Using a consistent framework provides the benefits of standardizing the developer workflow. Establishing consistency limits repeated work and saves developer time by removing the necessity to mull over the pros and cons of different industry tools. It also speeds up the code review process by making code more consistent, subsequently improving efficiency for future development on the project. Additionally, it allows for

```python
from transformers import ViTImageProcessor, ViTForImageClassification
from PIL import Image
import requests
from fastapi import FastAPI

processor = ViTImageProcessor.from_pretrained('google/vit-base-patch16-224')
model = ViTForImageClassification.from_pretrained('google/vit-base-patch16-224')

app = FastAPI()

@app.post("/classify_via_url/")
async def classify_via_url(image_url):
    image = Image.open(requests.get(image_url).raw)
    inputs = processor(images=image, return_tensors="pt")
    outputs = model(**inputs).logits
    prediction_id = logits.argmax(-1).item()
    label = model.config.id2label[prediction_id]
    return label
```

Fig. 3. An example API endpoint written with a FastAPI library.

a faster onboarding process for new team members. In our case, the project team is expected to grow and will include undergrad and graduate college interns.

C. Design Decisions

Initially, model endpoints were created using a different API framework, BentoML [14]. This tool is more attuned to AI/ML applications than FastAPI, and features containerization, model storage, and deployment tools. However, after creating several API endpoints, it was decided that the ML-specific benefits of utilizing this library did not outweigh the costs of learning and managing an additional framework. FastAPI was able to accomplish the same core tasks in our usage while also providing flexibility to include ML models not using standard frameworks. It also had wider support for non AI/ML specific tools such as logging and authentication. For the sake of consistency and speed of development, the BentoML endpoints were rewritten using FastAPI. This also provided flexibility for supporting emerging scientific AI/ML models as well as more traditional models.

The separation between the REST API layer and the API endpoints layer allows for the memory-intensive models to be loaded into memory ahead of time, saving time for requests. An algorithm to load and unload models from memory is currently being developed. This will ensure that energy is saved when users are not using the platform (i.e. at night), while balancing the desire for fast inference results when usage rates for specific models are higher.

A researcher seeking to use models can interact with the API through typical means. The API can be called in code via an HTTP request package or through user-facing applications. We included varied access methods to ensure that researchers at any level of technical knowledge will have convenient access to the API and can interact with it however they would expect.

Given data privacy concerns around the use of AI/ML on non INL resources, as well as standard IT security requirements, an additional requested functionality was for private access to models and training resources. This ruled out cloud resources for many tasks. Different levels of data privacy (public, INL-only, team-specific, invite-only) will be possible for each model via authentication using JSON Web Tokens. We intend to add means for researchers to specify these privilege levels themselves when they upload a model to the platform.

IV. Use Cases

An MLOps platform provides the opportunity for researchers to integrate machine learning into their post experiment processes. If a researcher wants to use a pre-trained model or model presented in a research paper, they are presented with dependencies challenges, varying computing environments, or inexperience with the frameworks in which the models may have been used. While some of these challenges can be solved by portability tools such as containers, this still presents challenges for those who are not familiar with container tools. This is unfortunate as the published research is a valuable resource. Furthermore, many models are customized such that the training or inference of the model is not standardized, which makes the use of many MLOPs platforms challenging. An MLOps platform alleviates some of these challenges by hosting these models. To demonstrate the capabilities, several models in the NLP, computer vision, and computer audio domains were used. Within these domains, several large language models (LLMs), speech-to-text (STT), image segmentation, and image labeling models were added to the hosting platform. Each of these examples can be run independently of the rest of the system as they are built into a FastAPI interface and containerized. The following sections provide examples of the models that were utilized.

A. Large Language Models

NLP models, especially LLMs, are advancing rapidly and can be used for tasks in nuclear energy research and nuclear power plants such as condition report documentation, system logging reviews, and analysis of safety reports [23]. Finely tuning models on domain specific or organizational data is becoming more common, however many of these models are executed from notebooks in specific computing environments or command line scripts. To make these models more accessible, API interfaces were created around LLMs to securely process data and generate results. Initially, only publicly available pre-trained LLMs have been integrated, however additional work is planned to train and implement domain specific models. Galactica [5] from Meta and Dolly [6] from Databricks were used. Both of these models have a Hugging Face repository, so loading the models via the transformers library is trivial. These LLMs had pipelines for question-answering as well as text generation.

B. Computer Audio

Many workflows may include inference on audio files. As a proof of concept to test inference with audio files, a STT model from Hugging Face was implemented. This presented the possibility for automatic translation, video transcriptions, or captions. Additionally, this would allow for future text processing to occur based on a verbal request, comparable to that which occurs with personal AI assistants. Future audio based AI/ML models may not use STT, however audio model capabilities may be used to detect abnormalities in nuclear power plants [26].

C. Image Segmentation

Many experiments generate images that need to be classified or segmented. To provide an example of this capability, two previously published transmission electron microscopy (TEM) models were utilized. The models use images from particle beam microscopes that visualize specimens and generate a highly-magnified images. The first model from Jacobs et al. [2] performs semantic segmentation in TEM images, while the second model from Shen et al. [3] segments defects in TEM images. These models were made available via Github or as a zip folder and hosted inside of a container for dependency management.

D. Computer Vision

The final set of hosted models includes basic object classification models, also from Hugging Face. While these pretrained models may not be particularly useful for many researchers, they do provide examples of what could be done with fine tuning or when hosting newly trained models. Future work aims to implement optical character recognition models and domain specific equipment classification models.

V. Summary and Future Work

While developing an initial MLOps system has merit, work remains to solve a wider AI/ML problem. Given the unique circumstances of INL's research environment, optimizing for ease of use, security, reliable hosting, reproducibility are all important factors. These specifications have led to outlining stages of maturity for the platform, which were used to educate choices of which features to include in this initial stage of the project. Additionally, the stages of maturity assisted in defining features needed in the future to mature the system to solve more AI/ML challenges.

MLOps maturity frameworks have been defined that can be used to determine the stage at which an organization is at in the process of adopting MLOps principles. According to [8] the stages are as follows, listed in order of ascending maturity: data collection, automated model deployment, and lastly, model monitoring. A company in the first stage of MLOps maturity has implemented software that collects data automatically from some source, such that no one needs to periodically grab that data manually. In the next stage, the company will have have created a system to use the updated data to train and deploy a new version of the model. At stage three, the company integrates monitoring of the autonomously-updating model, such that the system will notify the team of model drift when it is detected. In the final stage, detecting worsening performance of the model would then trigger a regression to a previous version of the model, and perform automated retraining and redeployment of the model to correct the error. At this final stage, protection against data attacks and other ML-specific security issues becomes increasingly important.

The stages of MLOps maturity for our platform were defined as follows:

1) No MLOps
   - Deployment and hosting are done from scratch for each model, if at all. To use a model, ML engineers and software developers need time and resources to set up a tool for the specific use-case. At this stage, ML is underutilized, because none of the principles of MLOps have been implemented.
2) Model Hosting
   - Once a model and hosting platform exist, adapting new models to run should be easier. There is an established system and workflow for hosting and deploying the model using best practices. CI/CD, various types of code testing, basic security features, and API access are all implemented, resulting in a minimal but functional end-to-end ML life cycle application.
3) MLOps and Usability
   - Once mature MLOps features are implemented. Research and ML-specific utilities such as data and model versioning are used. Additional accessibility tools such as dashboards for researchers to indepen-

dently upload models, control access level, and alter dynamic parameters can be created.

Before the creation of an MLOps platform, INL was in the first stage: to host a model, a researcher would have to figure out a method to host the model themselves, and little if any work could be reused when a second researcher set out to host their own model. More likely, the researcher would choose not to host or publicize the model at all, beyond publishing a paper about the model's capabilities. At that first stage of maturity, two models would have their weights saved on different storage systems or hosted at disparate web addresses, likely using different frameworks, with no cohesion between their user interfaces. Because of this, a researcher hoping to iterate on the work of another would face several hurdles to doing so.

With the creation of the first version of an MLOps platform, INL has jumped to the second stage of maturity. Models can be easily hosted through the platform, saving time and increasing the reproducibility, lifespan, and reach of custom ML models. A variety of powerful models are always ready to use. Additional work is planned to cement the second level of maturity and then move to the third level of maturity. At that point, additional features would include: offline inference, a dashboard for ML engineers to upload their models to the platform, model and data set versioning, as well as model drift detection, further security implementations, and subsequent automatic retraining where applicable. However, initial work on this MLOps platform for nuclear energy research has already removed some barriers to using ML in scientific workflows at INL. We have shown that MLOps principles are capable of promoting software-enabled discoveries while further development will expand the functionality and impact of the project to meet the original goals.

ACKNOWLEDGMENT

This research made use of the resources of the High Performance Computing Center at Idaho National Laboratory, which is supported by the Office of Nuclear Energy of the U.S. Department of Energy and the Nuclear Science User Facilities under Contract No. DE-AC07-05ID14517.

Presented at USRSE2023, Chicago, IL, October 16-18, 2023

REFERENCES

[1] S. J. Parker and M. W. Anderson. "Nuclear science user facilities high performance computing: FY 2022 annual report". United States: N. p., 2023. Web.

[2] R. Jacobs, et al. "Performance and limitations of deep learning semantic segmentation of multiple defects in transmission electron micrographs", Cell Reports Physical Science, Volume 3, Issue 5, 2022, 100876, ISSN 2666-3864, Web.

[3] M. Shen, et al. "A deep learning based automatic defect analysis framework for In-situ TEM ion irradiations", Computational Materials Science, Volume 197, 2021, 110560, ISSN 0927-0256, Web.

[4] A. Paleyes and R-G. Urma and N. D Lawrence. "Challenges in deploying machine learning: a survey of case studies", ACM Computing Surveys, 55(6), 1-29. 2022. Web.

[5] R. Taylor, et al. "Galactica: A large language model for science." arXiv preprint arXiv:2211.09085 (2022).

[6] M. Conover, et al. "Hello Dolly: Democratizing the magic of ChatGPT with open models." Databricks blog. March 24, 2023. Web.

[7] R. Chard et al. "Publishing and serving machine learning models with DLHub", Proceedings of the Practice and Experience in Advanced Research Computing on Rise of the Machines (learning), pp. 1–7, 2019, Web, in press.

[8] M. M. John, H. H. Olsson, H. Holmström, and J. Boschl, "Towards MLOps: A Framework and maturity model," 2021 47th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), 2021, Web, in press.

[9] D. Morgan, G. Pilania, A. Couet, B. P. Uberuaga, and C. Sun, "Machine learning in nuclear materials research," Current Opinion in Solid State and Materials Science, 100975, 2022, Web, in press.

[10] C. Worrell, L. Luangkesorn, J Haight, and T Congedo. "Machine learning of fire hazard model simulations for use in probabilistic safety assessments at nuclear power plants,", Reliability Engineering & System Safety, pp. 128–142, 2019. Web, in press.

[11] Y. Cui. "Machine learning in safeguards at pebble bed reactors," Brookhaven National Lab.(BNL), 1679954, October 2020, Web, in press.

[12] Z. Ma, H. Bao, S. Zhang, M. Xian, A. L. Mack. "Exploring advanced computational tools and techniques with artificial intelligence and machine learning in operating nuclear plants," Idaho National Lab.(INL), 2022. Web, in press.

[13] FastAPI, tiangolo, Sebastián Ramírez, https://github.com/tiangolo/fastapi, https://fastapi.tiangolo.com/

[14] BentoML, https://github.com/bentoml/BentoML, https://www.bentoml.com/

[15] A. Boehnlein, et al. "Colloquium: Machine learning in nuclear physics". Reviews of Modern Physics, Volume 94, number 3, pp 031003. 2022. Web.

[16] C. Tang, et al. "Deep learning in nuclear industry: A survey." Big Data Mining and Analytics 5.2 (2022): 140-160.

[17] Medel, V. et al. "Modelling performance & resource management in kubernetes". In Proceedings of the 9th International Conference on Utility and Cloud Computing. 257-262.

[18] Merkel, D. (2014). Docker: lightweight linux containers for consistent development and deployment. Linux j, 239(2), 2.

[19] Ebert, C., Gallardo, G., Hernantes, J., & Serrano, N. (2016). DevOps. Ieee Software, 33(3), 94-100.

[20] Jouppi, N. et al. "Tpu v4: An optically reconfigurable supercomputer for machine learning with hardware support for embeddings." In Proceedings of the 50th Annual International Symposium on Computer Architecture. 1-14.

[21] Milan, P. J. et al. "Enabling real-time adaptation of machine learning models at x-ray Free Electron Laser facilities with high-speed training optimized computational hardware". Frontiers in Physics, 982.

[22] M. Raissi, P. Perdikaris, G. Karniadakis. "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations". Journal of Computational physics, 2019. Volume 378, pages 686-707.

[23] A. Rashdan, B. Wilcken, K. Giraud. "An Artificial Intelligence-driven MIRACLE for Condition Reports Screening and Processing". Sept. 2022.

[24] M. Zhegang, et al. "Exploring Advanced Computational Tools and Techniques with Artificial Intelligence and Machine Learning in Operating Nuclear Plants." No. INL/EXT-21-61117-Rev001. Idaho National Lab.(INL), Idaho Falls, ID (United States), 2022.

[25] H. Tanuwidjaja, et al. "Privacy-preserving deep learning on machine learning as a service—a comprehensive survey." IEEE Access 8 (2020): 167425-167447.

[26] A. Marklund, J. Dufek. "Development and comparison of spectral methods for passive acoustic anomaly detection in nuclear power plants." Applied acoustics 83 (2014): 100-107.

[27] P. Plachinda, C. Ritter, and P. Sabharwall. "Digital Engineering Sensor Architectures for Future Microreactor Builds". United States: N. p., 2021. Web. doi:10.2172/1827623.