Publishing and Serving Machine Learning Models with DLHub

Conference Paper · July 2019

DOI: 10.1145/3332186.3332246

9 authors, including:

Ryan Chard
Argonne National Laboratory
35 PUBLICATIONS **312** CITATIONS
SEE PROFILE

Logan Ward
Argonne National Laboratory
53 PUBLICATIONS **657** CITATIONS
SEE PROFILE

Zhuozhao Li
University of Virginia
29 PUBLICATIONS **176** CITATIONS
SEE PROFILE

Yadu Babuji
University of Chicago and Argonne National Laboratory
19 PUBLICATIONS **74** CITATIONS
SEE PROFILE

Publishing and Serving Machine Learning Models with DLHub

Ryan Chard, Logan Ward, Zhuozhao Li, Yadu Babuji,
Anna Woodard, Steven Tuecke, Kyle Chard, Ben Blaiszik, and Ian Foster
Data Science and Learning Division, Argonne National Laboratory
Department of Computer Science, University of Chicago

ABSTRACT

In this paper we introduce the Data and Learning Hub for Science (DLHub). DLHub serves as a nexus for publishing, sharing, discovering, and reusing machine learning models. It provides a flexible publication platform that enables researchers to describe and deposit models by associating publication and model-specific metadata and assigning a persistent identifier for subsequent citation. DLHub also supports scalable model inference, allowing researchers to execute inference tasks using a distributed execution engine, containerized models, and Kubernetes. Here we describe DLHub and present four scientific use cases that illustrate how DLHub can be used to reliably, efficiently, and scalably integrate ML into scientific processes.

1 INTRODUCTION

Rapid growth in the volumes and variety of observational data and simulation output, plus the emergence of new data analysis methods, such as those based on various forms of deep learning, are producing remarkable new approaches to science. For example, researchers are using such methods to develop low-cost surrogates for expensive models [20], detect extreme events in datasets [21, 22], and inform real-time experimentation [30]. Machine learning (ML) models provide a low-cost and efficient tool to rapidly evaluate a search space, identify new experimental candidates, and process large swaths of data with high degrees of accuracy. The potentially revolutionary advantages of incorporating ML in the scientific process is driving the need to facilitate model-in-the-loop research. The model-in-the-loop paradigm is increasingly necessary to make large scale scientific discovery both practical and affordable. However, there is a general lack of support for deploying models for both low latency and high-assurance inference, capable of serving sporadic scientific demands.

Instead, many researchers resort to ad-hoc solutions, wrapping their models in custom Web services [8, 18] and repurpose code repositories for dissemination [27]. These custom solutions are both time consuming to create and error prone to operate. As ML becomes increasingly pervasive throughout almost the entirety of scientific fields, the enormous aggregate effort spent developing and operating these services in turn spirals out of control.

In addition, the widespread use, comparative study, and evolution of new ML methods is hindered by practical obstacles such as inaccessible code and data, inconsistent data formats, lack of documentation concerning implementation approaches, and steep learning curves for associated tools. The models and methods themselves may not be easily accessed, understood, or applied. An unfortunate consequence of these various difficulties is that a researcher who reads about a new method in a journal article can rarely apply that method to new data or adapt the method for a new purpose. In addition, scientific research presents a number of unique requirements that are not necessarily applicable in other domains. For example, it is important to associate citation information, provenance, and usage tracking with models such that researchers are given credit for their work.

We argue that these obstacles to scientific progress can be overcome by establishing a **Data and Learning Hub for Science** (DLHub) [15] that serves as a connection point between providers of data analysis and modeling methods and associated data, on the one hand, and consumers of those methods and data. For providers, DLHub makes it easy to package up and publish individual data processing models, pipelines linking multiple such models, and data used to test and train such models and pipelines. For consumers, DLHub makes it easy to discover previously published models, access associated test and training data, run models on both test data and new data, and adapt models for new purposes.

While other model repositories [6] and serving technologies [16, 24] have been developed, they fail to meet the need to facilitate the publication and dissemination of ML advancements within the scientific community. Instead, serving platforms focus on large scale serving for production use in business operations, such as recommendation systems for online shopping, while model repositories enable users to download and run models locally. DLHub provides a nexus for ML researchers. Enabling them to publish, find, and use models by combining rich cataloging capabilities, fine-grained access control, and high performance serving capabilities along with the necessary computing infrastructure required to use models and incorporate them in model-in-the-loop research pipelines.

In the remainder of this paper, we first describe DLHub, focusing on its architecture and capabilities, and briefly evaluate DLHub's serving performance. We then outline four scientific use cases that illustrate how DLHub is currently used in practice. Finally, we summarize our experiences.

2 DATA AND LEARNING HUB FOR SCIENCE

DLHub is comprised of two core components: a model repository and model serving system. DLHub's model repository allows users to publish, cite, discover, and reuse ML models from a variety of domains. DLHub collects rich metadata

DOI: 10.1145/3332186.3332246

and combines search capabilities to enable users to discover, access, and use published models. We leverage standard metadata schemas to describe models including, common publication metadata (e.g., author, title, date), descriptive ML metadata (e.g., model type), and performance metadata (e.g., accuracy when applied to common benchmarks). Model metadata is stored in a flexible search index, built on Globus Search [10], that allows users to query across all registered metadata. DLHub also allows users to associate a persistent identifier with a published model such that it can be cited by other researchers.

Each model published into DLHub is dynamically converted into a *servable*—an executable container that implements DLHub's standard execution interface. These containers include all of the dependencies necessary to both invoke the model as well as enable DLHub to serve the model for on-demand inference. DLHub's serving infrastructure provides a low-latency and scalable means of deploying and invoking models using elastic computing infrastructure. This serving infrastructure is built upon Parsl [12] and Kubernetes. When users submit inference requests, DLHub uses Parsl to provision execution containers on Kubernetes dynamically. Each container includes a Parsl worker that DLHub subsequently uses to manage the execution of inference tasks in that container.

2.1 Interfaces

DLHub implements Python SDK and CLI interfaces to make it easy for model practitioners to publish, share, and invoke models on-demand. Figure 1 illustrates the use of the SDK to assemble a description of a model, publish the model in DLHub, and invoke the model using an example from the CANDLE project [31]. Both the SDK and CLI provide a collection of helper functions to support the description of models. Our basic approach is to generate as much metadata as possible and require users only complete templates or particular metadata values. DLHub provides rich search capabilities that enable the discovery of accessible models using free-text and structured queries. Having discovered a model, users can easily invoke the model by using the SDK to marshal input data and call the model's *run* function. DLHub applies a fine grain access control model and tracks access and usage on a per-modal basis. This allows researchers to share models with a select group of users, or publicly if they desire. The search index enforces these permissions thereby restricting discovery to only those users permitted to view a model.

2.2 Model Definitions

Each servable is defined by a schema that captures its provenance, computational environment, and its interface. Our design for creating the schema was guided by two goals (1) maximizing reuse of existing data models, and (2) minimizing the learning curve for scientists unfamiliar with schemas. The servable definitions both make the servable discoverable and provide a recipe for the DLHub serving infrastructure to instantiate the servable.

We employ the DataCite [28] metadata schema to describe the provenance of each servable. The DataCite schema allows for capturing the authors, a human-understandable description of the application, and links to related artifacts (e.g., publications describing the model, training datasets).

The computational environment is described by the software dependencies and files required to execute a servable. Software dependencies are captured either by a user listing the Python package dependencies in the schema, or listing repo2docker [5] configuration files as dependencies. Files required by a servable are stored in the schema in a dictionary as named files for those with a specific purpose (e.g., the weights file of a Keras model) or simply must be present in the environment (e.g., license information).

The interface description portion of the DLHub schema includes information needed for humans to understand model inputs/outputs, and for the DLHub web service to correctly invoke the servable. The servable definition starts with the type of function being served (e.g., a Python class method) and information needed to load configuration files from disk (e.g., serialization method for a scikit-learn model). Each method supplied by the servable is defined by a human-friendly free-form description and a controlled description of the data types using a JSON-schema-like definition. The method descriptions may also include information needed by DLHub to construct the servable (e.g., module and name of Python function).

Much of the information in the servable definition can be extracted automatically. For example, the types and shapes of the input arrays to a Keras Model are stored within the HDF5 model file saved by Keras. As shown in Figure 1, the DLHub SDK reads all of this information from the HDF5 file so that creating the minimal servable metadata needed for a Keras model requires only 3 lines of code. We have built similar tools around generated DataCite-compatible metadata, defining the computational environment, and for describing different types of servables (e.g., TensorFlow, Scikit-Learn).

2.3 Publishing Servables

The metadata and files that comprise a servable can be published on DLHub via several routes. The most common method is to send the data to DLHub via HTTPS. During publication, the DLHub SDK packages all of the files listed in the servable description and sends them to the DLHub Web Service as a compressed archive. The web service then generates a servable according to the recipe defined in the servable description. First, a Docker container is created in accordance to the repo2docker files included in the servable description and is populated with the files included by the user. The container is then supplied with a "shim" appropriate to the type of servable (e.g., Keras Model) that is used to generate a Python object with all of the methods described in the servable description that can be invoked by the DLHub executor. The container is then published to Amazon Elastic

Figure 1: Illustrative uses of DLHub Python APIs, here used to (1) assemble a description of a model, with relevant metadata; (2) publish the new model to DLHub; and (3) invoke the published model.

Container Registry (ECR), where it readily-accessible by any of the task manages. DLHub will be soon extended to accept large models by downloading them via HTTP (e.g., from Amazon S3) or transfering them from Globus endpoints.

DLHub also provides a GitHub-based workflow for publishing models. In this model, users simply construct a GitHub repository with a set of JSON-based metadata files. These metadata files require only the standard DLHub servable description to be saved in the root directory. Users may then choose to register the repository in DLHub for subsequent use. DLHub uses repo2docker to construct the servable and follows the same registration process as described above.

2.4 Managing model serving

Figure 2 shows how DLHub's model serving infrastructure is used to execute inference tasks. DLHub separates the repository and model serving REST service from the set of execution resources used for inference. Execution resources are represented by a DLHub *Task Manager* that is responsible for deploying and managing servables deployed on computing infrastructure. The figure shows how models are served by connecting Task Managers with the DLHub Management Service via low-latency message queues. These message queues allow DLHub to transmit inference requests and input data, or data references, to the Kubernetes-based serving infrastructure. There, requests are routed into the appropriate servable using one of the available *executors*. Requests are typically served through a custom Parsl [12] executor; however, they can also be served through model-specific execution frameworks: SageMaker and TensorFlow Serving. Parsl provides a general-purpose method of executing arbitrary models via a low-latency and reliable execution model, which interfaces with Kubernetes and manages servable deployments, including scaling replicas. It also implements an inference cache to improve model inference performance. At present, we have deployed the Task Manager on a 14-node Kubernetes cluster, called PetrelKube, hosted at Argonne National Laboratory.

2.5 Security

DLHub's security model ensures all operations—from publication to inference—are authorized and tracked for auditing and accountability. DLHub relies on Globus Auth to provide federated authentication and a common authorization framework. Users can login with one of hundreds of supported identity providers (e.g., institutions, ORCID, Google). Once authenticated, the Management Service validates users' credentials and acquires short-term access tokens to perform authorized actions on their behalf. For example, the Management Service will use a token to obtain profile information about a user. It also uses a dependent token to access Globus Transfer on the user's behalf, thereby allowing DLHub to download models and data as requested by users. The DLHub service is also registered as a Globus Auth resource server with it's own scope. This allows developers to build upon DLHub via secure programmatic invocation of its REST APIs.

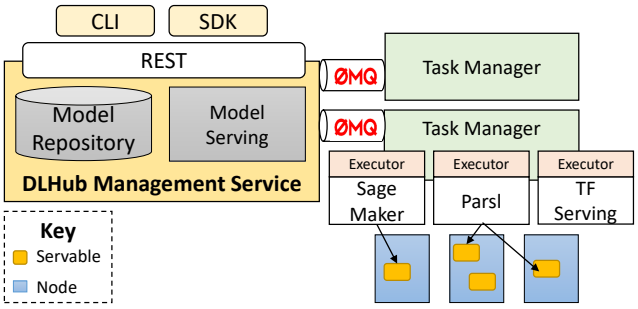

Figure 2: DLHub architecture. User requests, submitted via REST, SDK, or CLI (upper left) can result in model publication in the Repository or the dispatch of serving requests to servables deployed on any computing resources with a Task Manager interface and appropriate executor(s) (lower right).

3 EVALUATION

To evaluate DLHub's inference architecture we compare the scalability and latency performance of the two Parsl executors used in DLHub: IPyParallel (IPP) and HighThroughput (HTEX). The experiments were conducted on Argonne National Laboratory's PetrelKube, a 14-node Kubernetes cluster. Each node is equipped with two E5-2670 CPUs, 128GB RAM, two 300GB hard drives in RAID 1, two 800GB Intel P3700 NVMe SSDs, and 40GbE network interconnection. We use the IPP and HTEX executors to deploy a "no-op" servable, which returns "Hello World" when invoked.

To evaluate scalability we measure the completion time to process 10000 inference requests of the "no-op" servable. Figure 3 shows the completion time as the number of concurrently deployed servables is increased from 1 to 512. We observe that IPP has a much longer completion time than HTEX, and the maximum throughputs (defined as the number of processed tasks per second) for IPP and HTEX are 342 and 1531 tasks per second, respectively. In addition, the performance of both HTEX and IPP initially improves as more servables are deployed. However, the benefits diminish after 8 pods for HTEX and 4 pods for IPP. This is due to the short execution time of the "no-op" servable. Longer running servables typically benefit from more pods before experiencing diminished performance improvements. IPP's performance is further degraded with more than 128 pods, while HTEX's performance remains consistent even with a large number of pods. This is because the overheads associated with managing more nodes starts to outweigh the benefits of improved throughput.

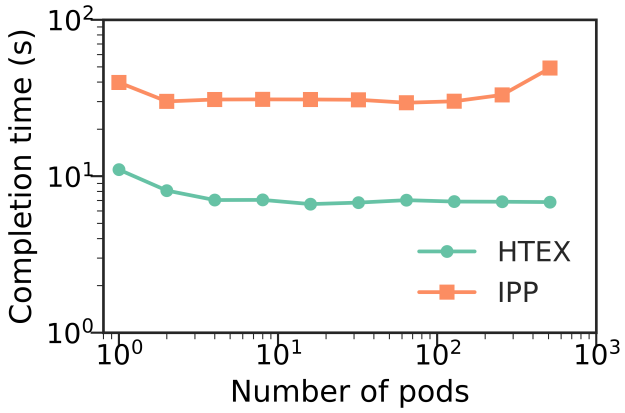

Figure 3: Scaling performance of IPP and HTEX.

We also measure the time for a single inference of the "no-op" servable using each of the executors to evaluate their latency properties. Figure 4 shows the latency distribution of 1000 repeated invocations. We observe that IPP has slightly lower latency for a single inference than HTEX (7 ms on average for IPP compared to 12 ms on average for HTEX).

In summary, IPP provides a slightly lower latency invocation model than HTEX, but is less scalable than HTEX.

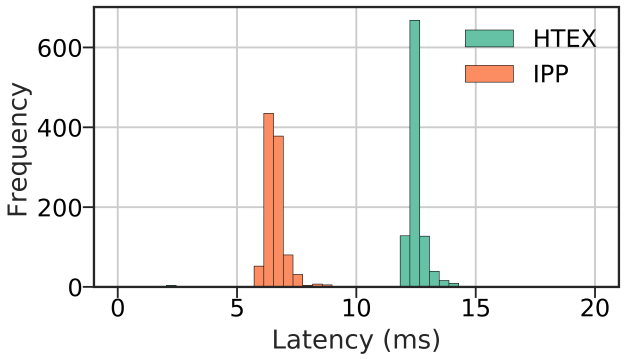

Figure 4: Latency performance of IPP and HTEX.

4 USE CASES

Initial usage of DLHub has focused on use cases at Argonne National Laboratory and in particular those associated with data stored in the Materials Data Facility (MDF) [13, 14]. We briefly highlight four scientific use cases currently supported by DLHub, including two related to materials science and one focused on cancer research.

4.1 Band Gap Prediction from Optical Images

A recent publication by Stein et al. [29] presents a model capable of predicting the material band gap and spectra from color optical images (64x64 pixels). The work describes a variational autoencoder (VAE) trained on 180,902 optical absorption spectra and optical images prepared via high-throughput experimental techniques. We analyzed these experimental absorption spectra using multiple adaptive regression splines (MARS) to locate the absorption onset to determine the material band gap. We then trained two models: Model 1) an autoencoder model implemented in Keras following the Stein example with latent space of size (100) to encode the optical images; and Model 2) a random forest regression model implemented in ScikitLearn and trained using the latent space from Model 1 as inputs and the band gaps determined by MARS as the outputs.

We have deposited the associated models into DLHub to encode an input image and output the latent space representation of the image. We have also deposited into DLHub the random forest regression model that can predict the band gap of the material given the latent space output. Together, these models allow researchers to submit an optical image from the Stein dataset to DLHub, encode the image and retrieve the latent space representation and subsequently use that output as input to the random forest regression model to obtain a predicted band gap. This use case exemplifies the ability for DLHub to enable researchers to validate published models, to apply them to their own data, and to build upon them by incorporating state-of-the-art ML techniques models into their research flows.

4.2 Bulk and Shear Moduli Model Benchmarking

It is typical that for any given materials property, such as melting point or band gap, there may exist many models in the literature that claim accurate predictive power. Often, these models are trained and evaluated on specific materials datasets, making it exceedingly difficult to effectively compare models. In addition, when these models are presented in literature, it is common for them to be discussed with regard to their model implementation or architecture, training sets, and their performance on a defined test set, yet rare for the trained model to be made publicly available. This makes it difficult to compare and contrast their capabilities.

DLHub provides a valuable tool for disseminating ML findings. Not only can authors publish their models and receive an associated persistent identifier they can include in their manuscripts, they can also share the model with others through the service. In addition, DLHub enables the direct comparison of models that perform similar tasks.

Figure 5 shows a direct comparison of different models derived from the literature. In particular, we examine the work from De Jong et al. [17] to predict the bulk and shear moduli from learned Hölder means descriptors. Similarly, we trained five models using different methods: linear regression, random forest regression, ridge regression, extra trees regression, gradient boosting, and an ensemble model. In this case, the ensemble model was the mean of each of the top 3 performing models (random forest regression, extra trees regression, and gradient boosting) by the validation mean absolute error. We use DLHub to deploy each of these models and then a simple Python script to invoke each model with the same input data and record their predictions. The figure demonstrates the different capabilities of the models, and enables users to make informed decisions regarding their accuracy. In addition, this example also shows that DLHub provides a novel means for performing ensemble predictions across multiple models.

4.3 Publication of Cancer Research Models

The Cancer Distributed Learning Environment (CANDLE) project [31] is designed to address cancer research problems at different biological scales. As part of the project researchers are developing and training models to study problems at the molecular, cellular, and population scales. To train these models, CANDLE leverages leadership scale computing resources at Argonne National Laboratory to rapidly perform hyperparameter optimization.

DLHub provides a method for the models developed in CANDLE to be securely published and served. For example, DLHub currently serves deep learning models and benchmarks that use cellular data to predict drug responses based on molecular features of tumor cells and drug descriptors. However, these models are still under development and require additional scrutiny from trusted collaborators before being made publicly available. DLHub provides an ideal mechanism to share these models with a selected group of users. Using DLHub's fine grain access control model, researchers can easily restrict access to a set of defined users and incrementally expose models to new group over times. Furthermore, when appropriate, users can make these models publicly accessible. DLHub's authorization model means that not only is model access restricted, but discovery is also restricted to the same, or different, levels of use.

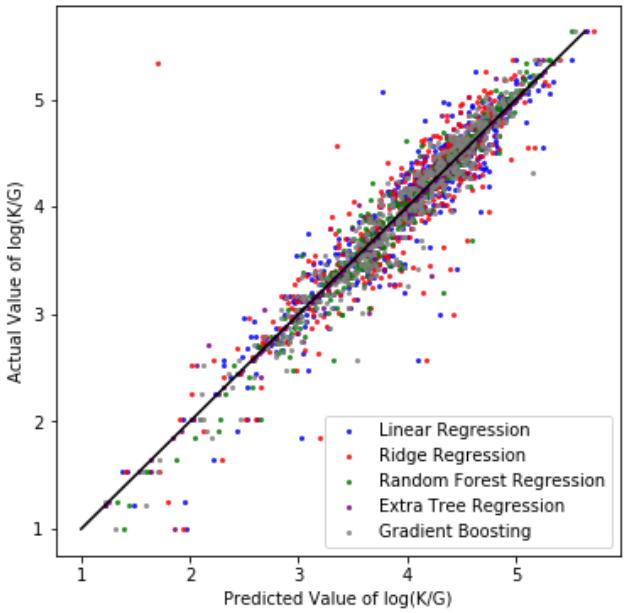

Figure 5: Using DLHub for prediction of bulk and shear moduli. Data are sent as inputs to five distinct models to evaluate and compare their performance.

4.4 TomoGAN

Recent work by Liu et al. [23] demonstrated a generative deep learning model, *TomoGAN*, that drastically improves 3D tomographs by reducing noise and eliminating artifacts. The key advantage of TomoGAN is that it allows for quality tomographs with only 1/16th of the normal X-ray dosage, either by imaging using fewer 2D projections of an object or with shorter exposure times. Improving the quality of reconstructions with *TomoGAN* makes it possible to study samples with fast dynamics or that are sensitive to X-ray damage. DLHub makes it possible to bring this capability to the general X-ray science community by hosting *TomoGAN* as a web service.

The *TomoGAN* model was created using TensorFlow, which means it only requires minor alterations to make it servable. DLHub reads the same SavedModel directory as TensorFlow Serving [25] and automatically generates a publishable model description from the files in that directory. As illustrated in Figure 6, users can send images to DLHub via HTTPS which will return a denoised image in less than a few minutes processing. By offloading the *TomoGAN* we can,

in the future, reduce the processing time by, for example, automatically parallelizing the evaluation of each frame or optimizing batch size — all without changing the user interface. In this way, DLHub will simplify deploying *TomoGAN* as a seamless part of X-ray tomography experiments.

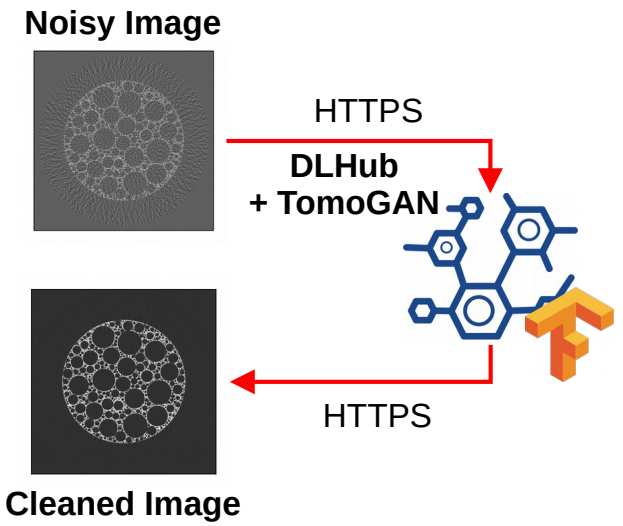

Figure 6: Using DLHub to use TomoGAN to eliminate noise from 2D projections.

5 RELATED WORK

Learning systems are defined as a system designed to support any phase of the ML model lifecycle, be it development [7], training [1], inference [16], or publication [3], to name a few. Learning systems are rapidly being developed and evolved to support the growing heterogeneity and pervasiveness of ML. Here we describe relevant serving and repository learning systems and compare their capabilities with DLHub.

Model serving platforms are essential to reliably and rapidly delivering ML inference on-demand. Various approaches have been taken to achieve ML serving, from hosted, cloud-based solutions, such as SageMaker [1], to self-service platforms such as Clipper [16]. SageMaker is a hosted platform provided by Amazon designed to aid users in selecting algorithms, developing models, training at scale, and deploying them for production use. Users can also export models as Docker containers for local use. Clipper on the other hand is a self-service platform, where users deploy it on their own infrastructure. Clipper is focused on low-latency invocation and provides several optimizations to improve serving performance, such as batching and memoization. However, both of these platforms rely on dockerized model containers to encapsulate model requirements, making them unsuitable for deployment on many HPC platforms. TensorFlow Serving [26] is arguably the most prominent inference platform. Using gRPC and REST APIs, TensorFlow Serving provides a high performance, low-latency solution for concurrently serving many ML models. Models are deployed by converting them to TensorFlow Servables. While some built-in transformations are supported, TensorFlow Serving does not support arbitrary transformation codes.

Model repositories present another area of learning systems where providers aim to deliver services for aggregating models and metadata such that users can discover and use models. This process is typically achieved by establishing a service to curate and publish model metadata and artifacts, such as ModelHub [3] and Kipoi [11], or by communities agreeing on a standard representation of a model [2]. DLHub aims to bridge these two areas of learning systems by facilitating self-publication of models and metadata while also providing low-latency serving to use the models on-demand.

Describing a model is essential to being able to discover, compare, and use them. Thus, model description tools are essential to the creation of model repositories and robustly deploying models in different platforms. The Open Neural Network eXchange (ONNX) [4] is one such description tool. ONNX is an open format for describing deep learning models that makes it possible to transform models between different tools (e.g., TensorFlow and Keras). Predictive Model Markup Language (PMML) [19] is an XML-based model description language that enables users and applications to exchange models and programmatically interpret their requirements and usage. Rather than focusing on describing the architecture and learned parameters of a model, the DLHub schema describes the purpose of the ML model, how to use it, and the environment details. In that way, DLHub is closer to a workflow/interface description language, such as CWL [9] or Google's protobuf. The DLHub schema also extends the interface description language by including DataCite-compatible provenance information to make models discoverable and useful to scientific research communities.

6 SUMMARY

Model-in-the-loop is a rapidly developing trend with wide ranging benefits including guiding experimentation, enhancing simulation campaigns, and ultimately reducing costs associated with research and accelerating discovery. In this paper we introduced DLHub, a service for publishing scientific machine learning models and enabling their discovery and use. We briefly described the DLHub architecture and highlighted the ease by which models can be published and used via the Python SDK. Finally, we described four use cases that currently use DLHub and benefit from its ability to make models discoverable, accessible, and servable. DLHub is accessible at dlhub.org.

ACKNOWLEDGMENTS

This research used resources of the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357. This work was also supported in part by NSF grant ACI-1148484. We gratefully acknowledge the computing resources provided

and operated by the Joint Laboratory for System Evaluation (JLSE) at Argonne National Laboratory.

REFERENCES

[1] [n. d.]. Amazon SageMaker. ([n. d.]). https://docs.aws.amazon.com/sagemaker/latest/dg/whatis.html. Accessed April 10, 2019.

[2] [n. d.]. Caffe Model Zoo. ([n. d.]). http://caffe.berkeleyvision.org/model_zoo.html. Accessed April 10, 2019.

[3] [n. d.]. ModelHub. ([n. d.]). http://modelhub.ai/. Accessed April 10, 2019.

[4] [n. d.]. ONNX. ([n. d.]). https://github.com/onnx. Accessed April 10, 2019.

[5] [n. d.]. repo2docker. ([n. d.]). https://repo2docker.readthedocs.io. Accessed April 10, 2019.

[6] 2019. ModelHub. (2019). http://modelhub.ai/. Accessed Febrary 20, 2019.

[7] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. TensorFlow: A system for large-scale machine learning. In *OSDI-16*. 265–283.

[8] Ankit Agrawal, Bryce Meredig, Chris Wolverton, Alok Choudhary, and Computer Science. 2016. A Formation Energy Predictor for Crystalline Materials Using Ensemble Data Mining. *Proceedings of IEEE International Conference on Data Mining (ICDM)* (2016), 1276–1279. https://doi.org/10.1109/ICDMW.2016.183

[9] Peter Amstutz, Michael R. Crusoe, Nebojša Tijanić, Brad Chapman, John Chilton, Michael Heuer, Andrey Kartashov, Dan Leehr, Hervé Ménager, Maya Nedeljkovich, and et al. 2016. Common Workflow Language, v1.0. (Jul 2016). https://doi.org/10.6084/m9.figshare.3115156.v2

[10] Rachana Ananthakrishnan, Ben Blaiszik, Kyle Chard, Ryan Chard, Brendan McCollam, Jim Pruyne, Stephen Rosen, Steven Tuecke, and Ian Foster. 2018. Globus Platform Services for Data Publication. In *Proceedings of the Practice and Experience on Advanced Research Computing (PEARC '18)*. ACM, New York, NY, USA, Article 14, 7 pages. https://doi.org/10.1145/3219104.3219127

[11] Ziga Avsec, Roman Kreuzhuber, Johnny Israeli, Nancy Xu, Jun Cheng, Avanti Shrikumar, Abhimanyu Banerjee, Daniel S Kim, Lara Urban, Anshul Kundaje, Oliver Stegle, and Julien Gagneur. 2018. Kipoi: Accelerating the community exchange and reuse of predictive models for genomics. *bioRxiv* 10.1101/375345 (2018).

[12] Yadu Babuji, Anna Woodard, Zhuozhao Li, Daniel Katz, Ben Clifford, Rohan Kumar, Lukasz Lacinski, Ryan Chard, Justin Wozniak, Ian Foster, Michael Wilde, and Kyle Chard. 2019. Parsl: Pervasive Parallel Programming in Python. In *ACM International Symposium on High-Performance Parallel and Distributed Computing*.

[13] Ben Blaiszik, Kyle Chard, Jim Pruyne, Rachana Ananthakrishnan, Steven Tuecke, and Ian Foster. 2016. The Materials Data Facility: Data Services to Advance Materials Science Research. *Journal of Materials* 68, 8 (2016), 2045–2052.

[14] Ben Blaiszik, Logan Ward, Marcus Schwarting, Ryan Chard, Jonathon Gaff, Daniel Evan Pike, Kyle Chard, and Ian Foster. 2019. A Data Ecosystem to Support Machine Learning in Materials Science. In *Materials Research Society, Special Issue Research Letter: Artificial Intelligence*.

[15] Ryan Chard, Zhuozhao Li, Kyle Chard, Logan T. Ward, Yadu N. Babuji, Anna Woodard, Steven Tuecke, Ben Blaiszik, Michael J. Franklin, and Ian T. Foster. 2019. DLHub: Model and data serving for science. In *33rd IEEE International Parallel and Distributed Processing Symposium*.

[16] Daniel Crankshaw, Xin Wang, Guilio Zhou, Michael J. Franklin, Joseph E. Gonzalez, and Ion Stoica. 2017. Clipper: A Low-Latency Online Prediction Serving System. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. 613–627.

[17] Maarten De Jong, Wei Chen, Thomas Angsten, Anubhav Jain, Randy Notestine, Anthony Gamst, Marcel Sluiter, Chaitanya Krishna Ande, Sybrand Van Der Zwaag, Jose J Plata, et al. 2015. Charting the complete elastic properties of inorganic crystalline compounds. *Scientific data* 2 (2015), 150009.

[18] Eric Gossett, Cormac Toher, Corey Oses, Olexandr Isayev, Fleur Legrain, Frisco Rose, Eva Zurek, Jesús Carrete, Natalio Mingo, Alexander Tropsha, and Stefano Curtarolo. 2018. AFLOW-ML: A RESTful API for machine-learning predictions of materials properties. *Computational Materials Science* 152 (sep 2018), 134–145. https://doi.org/10.1016/j.commatsci.2018.03.075 arXiv:1711.10744

[19] Alex Guazzelli, Michael Zeller, Wen-Ching Lin, Graham Williams, et al. 2009. PMML: An open standard for sharing models. *The R Journal* 1, 1 (2009), 60–65.

[20] Philip B Holden, Neil R Edwards, Paul H Garthwaite, and Richard D Wilkinson. 2015. Emulation and interpretation of high-dimensional climate model outputs. *Journal of Applied Statistics* 42, 9 (2015), 2038–2055.

[21] Thorsten Kurth, Jian Zhang, Nadathur Satish, Evan Racah, Ioannis Mitliagkas, Md. Mostofa Ali Patwary, Tareq Malas, Narayanan Sundaram, Wahid Bhimji, Mikhail Smorkalov, Jack Deslippe, Mikhail Shiryaev, Srinivas Sridharan, Prabhat, and Pradeep Dubey. 2017. Deep Learning at 15PF: Supervised and Semi-supervised Classification for Scientific Data. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '17)*. ACM, New York, NY, USA, Article 7, 11 pages. https://doi.org/10.1145/3126908.3126916

[22] Yunjie Liu, Evan Racah, Joaquin Correa, Amir Khosrowshahi, David Lavers, Kenneth Kunkel, Michael Wehner, William Collins, et al. 2016. Application of deep convolutional neural networks for detecting extreme weather in climate datasets. *arXiv preprint arXiv:1605.01156* (2016).

[23] Zhengchun Liu, Tekin Bicer, Rajkumar Kettimuthu, Doga Gursoy, Francesco De Carlo, and Ian Foster. 2019. TomoGAN: Low-Dose X-Ray Tomography with Generative Adversarial Networks. iii (2019), 1–17. arXiv:1902.07582 http://arxiv.org/abs/1902.07582

[24] Christopher Olston, Noah Fiedel, Kiril Gorovoy, Jeremiah Harmsen, Li Lao, Fangwei Li, Vinu Rajashekhar, Sukriti Ramesh, and Jordan Soyke. 2017. TensorFlow-Serving: Flexible, high-performance ML serving. In *31st Conf. on Neural Information Processing Systems*.

[25] Christopher Olston, Noah Fiedel, Kiril Gorovoy, Jeremiah Harmsen, Li Lao, Fangwei Li, Vinu Rajashekhar, Sukriti Ramesh, and Jordan Soyke. 2017. TensorFlow-Serving: Flexible, high-performance ML serving. *arXiv preprint arXiv:1712.06139* (2017).

[26] Christopher Olston, Noah Fiedel, Kiril Gorovoy, Jeremiah Harmsen, Li Lao, Fangwei Li, Vinu Rajashekhar, Sukriti Ramesh, and Jordan Soyke. 2017. TensorFlow-Serving: Flexible, high-performance ML serving. In *31st Conf. on Neural Information Processing Systems*.

[27] Fang Ren, Logan Ward, Travis Williams, Kevin J. Laws, Christopher Wolverton, Jason Hattrick-Simpers, and Apurva Mehta. 2018. Accelerated discovery of metallic glasses through iteration of machine learning and high-throughput experiments. *Science Advances* 4, 4 (apr 2018), eaaq1566. https://doi.org/10.1126/sciadv.aaq1566

[28] Joan Starr and Angela Gastl. 2011. isCitedBy: A Metadata Scheme for DataCite. *D-Lib Magazine* 17, 1/2 (jan 2011). https://doi.org/10.1045/january2011-starr

[29] Helge S Stein, Dan Guevarra, Paul F Newhouse, Edwin Soedarmadji, and John M Gregoire. 2019. Machine learning of optical properties of materials–predicting spectra from images and images from spectra. *Chemical Science* 10, 1 (2019), 47–55.

[30] B. Wang, K. Yager, D. Yu, and M. Hoai. 2017. X-Ray Scattering Image Classification Using Deep Learning. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*. 697–704. https://doi.org/10.1109/WACV.2017.83

[31] Justin M Wozniak, Rajeev Jain, Prasanna Balaprakash, Jonathan Ozik, Nicholson Collier, John Bauer, Fangfang Xia, Thomas Brettin, Rick Stevens, Jamaludin Mohd-Yusof, Cristina Garcia Cardona, Brian Van Essen, and Matthew Baughman. 2017. CANDLE/Supervisor: A Workflow Framework for Machine Learning Applied to Cancer Research. In *Computational Approaches for Cancer Workshop*.