Pangeo ML - Open Source Tools and Pipelines for Scalable Machine Learning Using NASA Earth Observation Data

PI:	Joseph Hamman	National Center for Atmospheric Research
Co-I/Institutional PI:	Ryan Abernathey	Columbia University
Co-I/Institutional PI:	David Hoese	University of Wisconsin, Madison
Co-I/Institutional PI:	James A. Bednar	Anaconda Inc.
Co-I:	Tom Augspurger	Anaconda Inc.

Contents

Summary

Consistent with the ACCESS Program's primary goal to adopt and implement technologies to effectively manage, discover, and use NASA's archive of Earth observations (EO) for scientific research and applications, the National Center for Atmospheric Research (NCAR) will lead the development of machine learning (ML) applications and open source technologies that meet specific computational needs of researchers and applied science practitioners. Over the course of a three year project, building on the Pangeo project, we will develop new high-level tools that serve a broad range of ML applications, primarily focusing on the extract-transform-load (ETL) pattern ubiquitous in ML workflows yet functionally unique to the geosciences.

Motivated by the idea that geoscientific ML workflows are unique in many respects (dimensionality, types of data transformations, and data volume), we plan to provide new tooling and resources to streamline the process of using EO data in deep-learning frameworks (e.g. TensorFlow). We will build on the robust open-source scientific Python ecosystem already familiar to scientists, the Pangeo Project's recent success in developing scalable cloud-based data analysis environments, and our collective experience providing guidance on ML best practices for geoscience (e.g. EarthML). As stated above, our primary focus will be on the pre-processing steps required to develop ML pipelines that use EO data, filling in key missing steps between the software libraries commonly used in geoscientific exploratory data analysis (e.g. Xarray, Dask, Intake) and the libraries commonly used for deep learning (e.g. TensorFlow, PyTorch). Planned development will improve our ability to easily combine datasets from multiple sources and provide high-level data pipeline tools for efficiently loading and processing batches of data for ML training and inference.

The development of these tools is motivated by pressing research questions that seek to integrate NASA EO data in cutting-edge machine learning applications. Our proposal includes two science applications to help focus effort, one motivated by the SWOT mission and another that aims to improve the skill of macroscale hydrologic models. The tools themselves will provide much-needed functionality to the open-source software ecosystem and enable ML applications across the geosciences. Furthermore, these developments will provide key improvements in our ability to share, reproduce, and scale ML workflows in the geosciences. Our proposed project will provide clear links between the ETL workflow tools developed under this effort and other ACCESS Program elements including data analytics in the cloud, analytics-optimized data stores, and high-value open-source science tools.

Our team is led by domain scientists with extensive experience using NASA EO data and applying machine-learning techniques, and by software developers with broad experience in the development of open-source software for data manipulation, machine learning, cloud computing, and big-data analysis. Our project will enable real-world machine-learning applications that draw on EO data from multiple NASA missions (e.g. SWAT, Jason, MODIS, Landsat) as well as various *in-situ* and model-generated datasets.

In summary, our project will develop flexible, open-source ML workflow tools to make it easier for scientists to develop cutting-edge applications that take advantage of NASA EO data.

1 Motivation and Background

These are exciting times in Earth System Science (ESS) research. The confluence of an unprecedented amount of data—sourced from in-situ observations, remote sensing platforms, and simulation models—and a wave of new algorithms that can effectively learn from this data promises to aid scientists and practitioners in unlocking new scientific discoveries and confronting pressing socio-environmental challenges [Reichstein et al., 2019]. Our proposal aims to catalyze this confluence through a) the development of open source software (OSS) designed to accelerate the deployment of a wide range of machine learning (ML) workflows, and b) the production of high-value demonstrations that address pressing research questions in multiple research domains.

There is no one-size-fits-all workflow for ML in the geosciences. Every ML application is made up of a unique combination of input datasets, preprocessing steps, model architectures, and scientific objectives. Over the course of the last year, and in conjunction with the Pangeo ML Working Group [Abernathey et al., 2019], our project team has concluded that there is a true need for foundational tools that facilitate the construction of the types of ML workflows commonly encountered in the geoscientists. In response to these conclusions, we propose focusing our development efforts in the following four areas:

1. Improvement, maintenance, and support of Pangeo software,
2. Expanded interoperability of the scientific Python ecosystem (e.g. Pangeo and Pytroll) to simplify construction of preprocessing pipelines for ML applications,
3. Development of new software interfaces between Xarray and machine libraries (e.g. Scikit-Learn, TensorFlow, PyTorch), and
4. Expansion of the EarthML open-source documentation platform for ML applications in the geosciences, capturing and disseminating the approaches developed here.

The Pangeo Project is a dynamic open source community collaboration with a primary mission of promoting open, reproducible, and scalable science. The project has pioneered the use of cloud computing for interactive scientific analysis at scale, developed new cloud-optimized data formats for multi-dimensional arrays (e.g. Zarr), and has helped foster an integrated open-source software ecosystem that has found broad applications well beyond the geosciences. Today, the Pangeo community has grown to include a diverse mix of domain scientists, software engineers, system administrators, and data providers. Informed by our experience developing the Pangeo Project and our recent experience within the project's ML working group, our proposal represents an exciting and reliable opportunity to grow the Earth science community's technical capacity in the subject area of machine learning.

1.1 Impact and Relevance to ACCESS Objectives

Our project builds on the successful technology and community development activities of the Pangeo Project over the past three years. Under support from the NASA ACCESS (80NSSC18M0156-PIs Hamman/Augspurger) and NSF EarthCube (174064-PIs Abernathey/Hamman/Bednar;

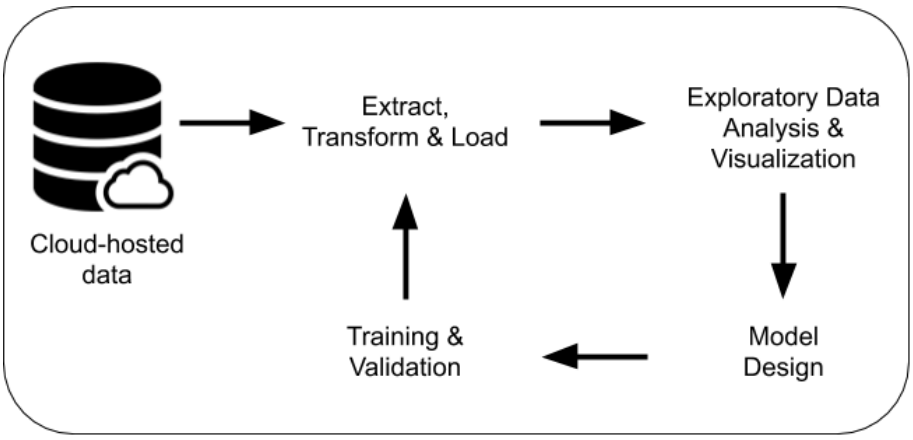

Figure 1: Hypothetical research workflow for a scientist using Pangeo-ML. Data hosted on the cloud is integrated into a familiar ecosystem that can provide extract-transform-load functionality as well as exploratory data analysis and visualization. Using the same tool sets, scientists will be able to quickly iterate on model design, and training and validation.

1928374 - PI Hamman) programs, Pangeo has developed a transformative approach to interactive exploratory data analysis at scale. Following the same model of community engagement and open-source development [e.g. Hamman, 2019], our proposal aims to extend this success to applications of machine learning in the geosciences by making key improvements in our ability to develop, iterate on, share, reproduce, and scale ML workflows.

Our proposal is uniquely aligned with the ACCESS Program's overarching goal to "adopt and implement technologies to effectively manage, discover, and use NASA's archive of Earth observations", simultaneously responding to the primary and secondary program elements. Our contribution to the first element falls squarely under the objective of addressing challenges faced by users interested in applying ML techniques to NASA data. Furthermore, our focus on, and demonstrated commitment to, improving and extending critical components of the open-source scientific Python ecosystem uniquely addresses the second program element, which aims at providing support to existing widely used open-source tools and libraries.

1.2 Machine Learning Patterns with Earth Observation Data

The Earth Science community makes use of many of the same open-source machine-learning tools that have become widespread in commercial applications over the past decade. The capabilities of these tools (TensorFlow, PyTorch, Keras, Scikit-Learn, and many others) have been advancing rapidly due to major investments from companies focused on specific commercial applications, and by machine-learning researchers looking to improve performance against benchmark datasets. However, typical commercial and academic applications differ in important ways from those in the Earth Sciences, where the data involved tend to be large, continuous, multidimensional geo-referenced datasets, rather than collections of isolated examples like individual images, documents, or transactions. To make use of general-purpose ML tools, Earth-science researchers often end up spending substantial time on the Extract-Transform-Load

(ETL) steps of a data-processing pipeline, slicing and reshaping the source data so that it fits the assumptions and requirements of the machine-learning tools available, and transforming the outputs of the machine-learning tools back into an interpretable geo-referenced form. Published studies have shown that data scientists in general spend only 20% of their time on actual data analysis, with the remaining 80% spent on ETL [Miller and Hughes, 2017]; based on our personal experience, we estimate the ratio for Earth Scientists is closer to 10-90, for the reasons described above.

This project focuses on simplifying, optimizing, and easing the flow of data from the general-purpose tools already in wide use for exploratory data analysis in Earth science using Python (Xarray, Dask, etc.) to and from the machine-learning tools, helping avoid having each Earth scientist create complex custom software to address these "impedance mismatches" that otherwise dominate the practical application of machine learning to Earth-science problems. The ultimate goal is to make it simple and straightforward for scientists to make end-to-end ETL pipelines starting with the libraries they already use for exploratory data analysis, while making use of the power available in modern machine-learning tools (Figure 1).

1.3 The Open Source Building Blocks for Pangeo-ML

The Pangeo Project has worked to foster an integrated ecosystem of software projects that support the analysis of large geoscientific datasets. Thus, a Pangeo environment is made of up of many different open-source software packages that are connected through common data structures and APIs. Because the Pangeo software ecosystem makes up the foundation of our proposal, we describe a few key projects in detail.

1. **Xarray** (1.3.1) provides intuitive computation on geospatial datasets.
2. **Dask** (1.3.2) sits beneath Xarray and provides parallel and distributed computing.
3. **Intake** (1.3.3) is a library providing access to data catalogs, search, and discovery.
4. **Pytroll** (1.3.4-5) is a collection of high-level tools for accessing, reprojecting, and remote sensing data.

An inevitable byproduct of the work proposed here is the continued general maintenance and development of these high-value tools and libraries, with broad benefits for the Earth Science community and beyond.

1.3.1 Xarray

Xarray is a community-developed, open-source software project and Python package that provides tools and data structures for working with multidimensional labeled arrays [Hoyer and Hamman, 2017]. The Xarray data model is based on the Common Data Model (CDM) used by NetCDF [Rew and Davis, 1990], which provides a standard for metadata-enabled self-describing scientific datasets. The labels used by Xarray come from the metadata described by the CDM. Labels (also referred to as coordinates) include quantities like latitude, longitude, time, and experiment number. Figure 2 illustrates an example Xarray dataset. PIs Hamman

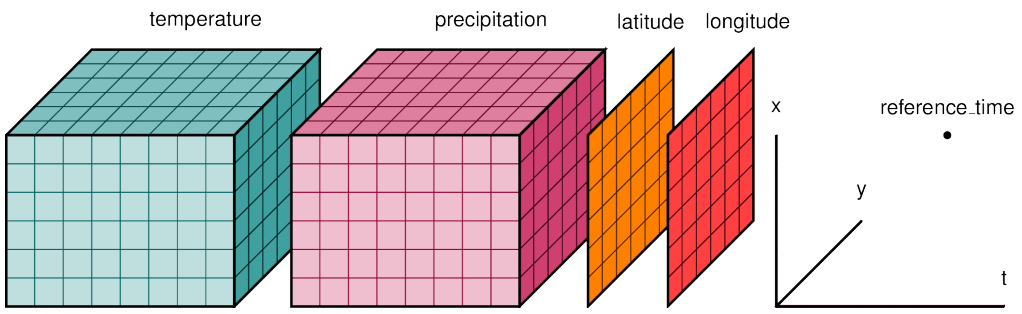

Figure 2: An example of how a dataset (NetCDF or Xarray) for a weather forecast might be structured. This dataset has three dimensions, time, y, and x, each of which is also a one-dimensional coordinate. Temperature and precipitation are three-dimensional data variables. Also included in the dataset are two-dimensional coordinates latitude and longitude, having dimensions y and x, and reference time, a zero-dimensional (scalar) coordinate. Figure and caption from Hoyer and Hamman [2017].

and Abernathey, are core developers on the Xarray project. Several Anaconda developers have also contributed to Xarray as well as maintaining packages based on or supporting Xarray (e.g. Pandas, Datashader, HoloViews, GeoViews, and Intake-xarray).

The development of Xarray has been led by the geoscientific computing community, and has been motivated by the lack of existing expressive, extensible, and scalable analysis tools that can be applied to modern geoscientific datasets. Xarray provides an interface designed to meet three primary goals: (1) Provide data structures that include the necessary metadata to facilitate label-based operations and eliminate common data-coding mistakes that arise from detaching datasets from the metadata that describes them; (2) Facilitate rapid analysis of array-based datasets by providing a high-level data analysis toolkit; and (3) Interoperate with a range of existing packages in and out of the geoscience domain.

Built on top of the Xarray data model is a robust toolkit that includes (1) label-based indexing and arithmetic; (2) interoperability with the core scientific Python packages (e.g., Pandas, NumPy, Matplotlib); (3) out-of-core computation on datasets that do not fit into memory (via Dask, see Section 1.3.2); (4) a wide range of serialization and input/output (I/O) options such as NetCDF, OPeNDAP, GRIB, HDF, Zarr, and various raster formats (e.g. GeoTIFF); and (5) advanced multi-dimensional data-manipulation tools such as group-by, resampling, and rolling window operations. Xarray also provides a plugin system referred to as "accessors" for adding domain specific functionality on top of core Xarray objects (DataArray and Dataset).

Xarray's high-level interface is well-documented, intuitive, and easy to use, even for those new to Python. For the geoscience community, these features allow relatively painless transitions to Python and Xarray. Once using Xarray, geoscientists are able to quickly produce meaningful scientific analysis without (1) writing excessive boilerplate code (e.g. for reading/writing datasets) or (2) being bogged down by imprudent reliance on a particular metadata structure. Furthermore, its expressive API leverages available metadata to enable a more intuitive, more concise, and less error-prone developer experience. We see Xarray as a *lingua franca* of the software ecosystem, enabling different packages to share rich data structures and metadata

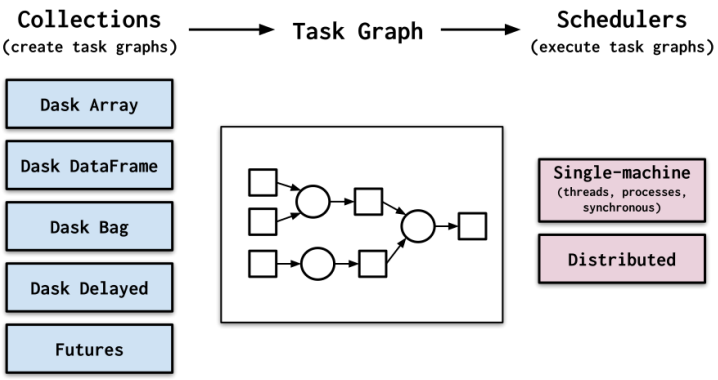

Figure 3: Dask is composed of two parts: 1) "Big Data Collections" like parallel arrays and dataframes and 2) "Dynamic Task Scheduling" that optimizes computation.

without resorting to writing files.

While Xarray is a highly capable general-purpose analytics library, it does not include machine learning capabilities or extensive geoscience domain knowledge. One of the primary focus points in this proposal is to improve the connections between Xarray, data-manipulation libraries that target remote sensing data (e.g. Satpy), and ML libraries (e.g. Scikit-learn, TensorFlow, PyTorch).

1.3.2 Dask

Dask is a system for parallel computing that coordinates well with Python's existing scientific software ecosystem, including libraries like NumPy, Pandas, and Scikit-Learn [Rocklin, 2015, Dask Development Team, 2016]. Data analysis tools like Xarray use Dask to parallelize complex workloads across many cores on a single workstation or across many machines in a distributed cluster. Dask manages running tasks on different workers, managing failed nodes, tracking the location of intermediate results, moving data across the network, collating and returning final results, etc. The Anaconda Co-I Tom Augspurger is a core developer on the Dask project, and PIs Bednar, Abernathey, and Hamman manage projects built on Dask (Datashader, Xarray, Dask-deploy).

Dask's high-level interface provides a familiar API to users of Python libraries like NumPy, pandas, and Xarray (Figure 3). Operations using the high-level interface generate a low-level *task graph*, which represents the computation written by the user. This task graph is then executed in parallel using many threads or processes on a single machine or many machines in a distributed cluster. Typical geophysical computations create complex dependencies between chunks of data. The Dask task system is flexible enough to represent these computations and execute them in an scalable, high-performance, adaptive manor.

Dask-ML builds on top of Dask to provide tools for scalable machine learning [Dask-ML Development Team, 2019]. It works with software packages like Scikit-Learn and XGBoost to scale machine learning workflows to larger-than-memory datasets stored in Dask Arrays and DataFrames. By following the established Scikit-Learn interface, Dask-ML can be easily included

in existing workflows and integrates nicely with other projects following the Scikit-Learn interface.

1.3.3 Intake

Intake is a lightweight Python package for describing, discovering, exploring, loading, and disseminating data. Pangeo applications commonly use Intake as a high-level data broker, providing functionality to both discover and load the complicated high-dimensional datasets commonly found in remote sensing or Earth system model data archives. In short, Intake lets researchers capture and share the metadata needed to locate and load whole categories of data, thereby avoiding having to write custom software code each time they need to access a particular piece of data. Under current support from the NASA ACCESS program, Intake support has been added for data indexed in NASA's Common Metadata Repository through a SpatioTemporal Asset Catalog (STAC) proxy [Hamman, 2019, Element 84, 2019], enabling convenient access to data whose metadata has been provided in a language-agnostic format. An example of using Intake-STAC to discover Landsat imagery and return Xarray datasets is shown in Listing 1.

```
import satsearch
from intake import open_stac_item_collection as open_stac

# use sat search to query a dynamic STAC API
properties = ["eo:row=027", "eo:column=047", "landsat:tier=T1"]
results = satsearch.Search.search(
    collection='landsat-8-l1',
    sort=['<datetime'], #earliest scene first
    property=properties)

sceneid = 'LC80470272019096'
catalog = open_stac(results.items())  # convert the satsearch results to intake
da = catalog[sceneid]['B2'].to_dask()  # Return an xarray dataset backed by dask
```

Listing 1: A basic usage example for Intake-STAC to search a dynamic STAC API and return an Intake Catalog that can produce data in the form of Xarray/Dask datasets.

1.3.4 Pyresample

Pyresample is a Python library for resampling geospatial image data between various projected coordinate reference systems (CRS). It is one of the tools developed by the open source Pytroll community and uses the Xarray data model. Pyresample is used primarily by the meteorological satellite community to reproject, subsample, or crop data to suit a particular use case. Users can choose from various resampling algorithms such as nearest neighbor or elliptical weighted average (EWA). Pyresample offers support for performing operations across multiple cores and the ability to cache intermediate resampling results which greatly improves performance for future execution. Various utilities are also available in Pyresample to help users working with geolocated data. These include helper functions for plotting data with the Cartopy library, computing bounding coordinates of a set of data, and determining intersections between two datasets.

1.3.5 Satpy

Satpy is a high-level library for working with satellite data and is another of the tools created by the Pytroll community. Like Pyresample, Satpy also uses the Xarray data model and Dask package for parallel computation. It allows users to read data from over 40 different satellite instrument file formats, manipulate the data by applying atmospheric and other corrections, resample data using Pyresample, and write data to disk in various formats. Satpy provides simple interfaces to make working with satellite data more accessible to scientists with little Python experience. By using the same interfaces regardless of satellite instrument, users are able to quickly transition their work from one data source to another.

2 Technical Plan

The central goal of our project is to enable cloud-native, scalable, and interactive deep learning workflows that use NASA EO data, by extending the Pangeo software stack to cover these use cases. Acknowledging that ML applications in the ESS community take many forms, our technical focus is to develop foundational tools that can be used to compose a variety of ML workflows. Here we summarize the four high-level areas of work in our technical plan.

1. **Improvement, Maintenance, and Support of Tools and Libraries Critical for Earth-Science Applications.** As we described in section 1.3, the Pangeo project provides an integrated ecosystem of many different open-source software packages. Our project team has demonstrated experience in developing, maintaining, and supporting many of these open-source tools. Under support of this program, we will continue to maintain and develop libraries critical for modern Earth-science applications, like Xarray, Dask, Intake, HoloViz, Satpy, and Pyresample.

2. **Improve integration of data-manipulation libraries.** Central to the machine-learning scientist's toolbox is a suite of open-source software libraries for data manipulation. These libraries (e.g. Xarray, Pyresample, and Intake) are key to the exploratory data analysis as well as the ETL steps in the ML workflows. Our team has extensive experience developing and maintaining these software projects, as part the Pangeo and Pytroll projects, and will use this project to further the integration of this ecosystem to better support ML applications.

3. **Develop software interfaces between Xarray and deep-learning libraries.** The Xarray library provides a data model for N-dimensional labeled arrays in Python and a powerful toolkit for expressive data manipulation. As Xarray is increasingly becoming a central tool for a wide variety of data processing applications in the geosciences, there is a need to extend the functionality provided by the core Xarray library to support the wide variety of ML workflows scientists are pursuing as they use NASA EO data. As we describe in section 2.2, we will develop an Xarray extension library (Xbatcher) to interface large, distributed, multi-dimensional Earth observation datasets with deep-learning libraries (e.g. TensorFlow, PyTorch).

4. **Provide documentation and demonstration.** Today's scientists are facing an onslaught of new technologies and methodologies. Migration to unfamiliar computing environments (e.g. the cloud) and the ever-evolving software environment often stunt the potential increase in productivity these technological advancements offer. One aspect of the solution to this challenge comes through high-quality documentation and interactive demonstration. To this end, we will extend the EarthML Project (https://earthml.holoviz.org/) by adding a collection of online interactive deep-learning tutorials for Earth science topics, based on the approaches developed and optimized elsewhere in this project.

In addition to our commitment to supporting the core open-source libraries in the Pangeo ecosystem (item 1 above), we will focus on three areas of development that will enable more productive ML workflows that use NASA data:

1. **Pytroll** (2.1): a suite of tools for processing Earth observation satellite data.
2. **Xbatcher** (2.2): a prototype library that produces batches of Xarray datasets for deep-learning libraries.
3. **EarthML** (2.3): a tutorial-based website that shows how to use open-source tools from the Python ecosystem to solve problems in earth science and climate science, focusing on machine learning and other automated analysis techniques.

2.1 Pangeo-Pytroll Integration

Pytroll is an open-source community that develops tools for working with Earth observation satellite data. These data constitute one of the primary inputs for ML applications. Many of the Pytroll tools already take advantage of Pangeo packages like Xarray and Dask for cleaner interfaces and increased performance. The integration of Pangeo projects into Pytroll tools like Pyresample and Satpy will be expanded as part of this proposal to serve a wider audience and benefit the Python scientific community as a whole, and the satellite-imagery ML community in particular.

Pyresample Initial development of Pyresample, which began years before libraries like Xarray and Dask existed, focused on supporting data sets that fit into memory (via NumPy arrays). While basic support for these libraries has been added on as needed, improved compatibility will allow for easier transformation and comparison between various data sets. To better support Pangeo-ML applications we will add a more formal interface for accessing Pyresample's features that is familiar to Xarray users via an Xarray extension. Pyresample's Dask support will also be improved so it can be used in more parallel-computing environments and Dask's multi-node scalability. Simple interfaces to parallel transformation algorithms can assist scientists in preparing their data for more complex analysis. Reprojection and alignment of images is frequently an expensive but necessary pre-processing step for ML applications with satellite imagery.

Satpy Xarray and Dask libraries are already important parts of Satpy, but like Pyresample it was originally developed for use with data sets that were small enough to fit into memory. Defining a formal Xarray extension for Satpy would simplify its use for users familiar with Xarray. Satpy will also benefit from integration with the Intake library to streamline the loading of satellite-imagery. Satpy can take advantage of Intake to access data from an Intake catalog, but Intake can also take advantage of Satpy for reading non-standard file formats.

2.2 Xbatcher Extension to Deep Learning Frameworks

A significant portion of the development effort in our proposal will focus on the development of the Xbatcher project. Xbatcher is a Python library for iterating through Xarray datasets in batches (the bite-sized chunks of data used in training ML algorithms iteratively). The goal of Xbatcher is to make it easy to feed Xarray datasets to the common machine learning libraries that train in batches (i.e. partial fit, gradient descent). The core functionality in the current version of Xbatcher is shown in Listing 2.

```
import xarray as xr
import xbatcher as xb

da = xr.open_dataset(filename, chunks=chunks)      # open a dataset and use dask
da_train = preprocess(ds)                          # perform some preprocessing
bgen = xb.BatchGenerator(da_train, {'time': 10})   # create a generator

for batch in bgen:                                 # iterate through the generator
    plt.scatter(batch['x'], batch['y'])            # explore the data batch-by-batch
    # or
    model.fit(batch['x'], batch['y'])              # fit a deep-learning model
    # or
    model.predict(batch['x'])                      # make one batch of predictions
```

Listing 2: A basic usage example for Xbatcher. Definition of the `filename`, `chunks`, and `model` variables and the `preprocess` function are omitted for brevity.

Xbatcher is still in an early prototype stage. While it is positioned to fill an exciting space in the Pangeo ecosystem, its development roadmap includes many exciting features not yet implemented. A sampling of the high-level features we plan to develop as part of this project are listed below:

1. **Plugin interface to Xarray**. Defining a plugin interface to Xarray (technically referred to as an "accessor extension") will facilitate the seamless integration of Xbatcher with existing Xarray workflows.
2. **Utilities for Shuffling and Sampling**. When training machine-learning models in batches, it is often necessary to selectively or randomly sample from your training data. The Pangeo ML working group has documented numerous challenges related to shuffling in geoscience ML workflows. We will identify best practices for shuffling via real science use cases and implement them in Xbatcher via a high-level expressive API. For large datasets this will involve implementing fast, distributed shuffles on Dask arrays.

3. **Integration with TensorFlow and PyTorch Dataset Loaders**. Deep-learning libraries like TensorFlow and PyTorch provide high-performance dataset-generator APIs that facilitate the construction of flexible and efficient input pipelines. In particular, they have been optimized to support asynchronous data loading and training, transfer to and from GPUs, and batch caching. Xbatcher will provide compatible dataset APIs that allow users to pass Xarray datasets directly to deep-learning frameworks.
4. **Dask Integration for Parallel Asynchronous Loading of Batches and Training**. The TensorFlow and PyTorch dataset APIs provide basic utilities for parallel dataset loading [e.g. Abadi et al., 2015]. We will work to extend this functionality to use Dask to support preprocessing on multiple machines and overlapping data loading and model training. This work will also make it possible to use Dask to coordinate distributed training for very large datasets.
5. **Pluggable Cache Mechanism**. A common pattern in ML is perform the ETL tasks once before saving the results to a local file system. This is an effective approach for speeding up dataset loading during training but comes with numerous downsides (i.e. requires sufficient file space, breaks workflow continuity, etc.). We propose the development of a pluggable cache mechanism in Xbatcher that would help address these downsides while providing improved performance during model training and inference. For example, this pluggable cache mechanism may allow choosing between multiple cache types, such as an LRU in-memory cache, a Zarr filesystem or S3 cache, or a Redis database cache.

2.3 EarthML

EarthML is a project developed by Anaconda in a previous non-Pangeo NASA Goddard SBIR contract, with the goal of making machine-learning and related tools more accessible to Earth-science researchers. In the previous project, best-practice approaches were developed and established for capturing, documenting, and publishing live, runnable, and fully reproducible tutorials and examples of complex cloud-friendly, scalable data-analysis and processing workflows. The current EarthML site includes example machine-learning and related workflows illustrating the use of Scientific Python tools like Xarray, Pandas, Dask, Dask-ML, Scikit-learn, Intake, and HoloViz, all of which are heaviliy used by the Pangeo community. The process of developing and improving these examples highlighted the need for small changes, fixes, and improvements to nearly all of these tools, which Anaconda was able to develop and contribute back to the underlying OSS libraries so that the improvements are now available for all users of those tools.

The existing EarthML examples are all cloud friendly using the same client-server approaches employed by Pangeo, but the current examples do not specifically cover how to scale these workflows for large cloud-hosted datasets, how to manage distributed computation, and how to work with climate simulation data or large collections of satellite images. In this project, a new suite of examples will be added to EarthML to capture and disseminate these Pangeo-focused

use cases, demonstrating how to create a reproducible recipe for cloud-based computation on NASA datasets and simulation data as outlined in the sections above. We will source these new examples from the Pangeo ML community and from the science application propopsed as part of this project. Just as in the previous (SBIR-funded) project, developing and optimizing these examples will undoubtedly suggest areas where each of the tools could be improved, and where feasible in the funded time available these improvements will be made directly on the underlying projects and contributed back to them for all users. This project will thus directly improve the scientific workflows it addresses (through improvements to the tools already in wide use), while capturing and disseminating best-practice and getting-started guides to allow all Earth scientists to make full use of these tools.

3 Science Plan

The technical objectives described above are motivated by pressing research infrastructure needs in active research projects lead by team members in the areas of oceanography (co-I Abernathey) and hydrology (PI Hamman). We detail a few of these ongoing research efforts in an effort to characterize the breadth of applications these tools will support. These use cases will leverage many of the open-source libraries described above, including Xarray, Dask, Pyresample, and Xbatcher. In section 4.1, we further describe how we plan to develop collaborations across the ML application space to amplify our project's impact.

3.1 Ocean Interpolation and Extrapolation (Abernathey)

Ocean observations, from both satellite and *in-situ* instruments, are typically sparse and unstructured in space and time. Scientists studying global climate variability, however, generally desire data on regular spatiotemporal grids (i.e. Level 3 / Level 4 products). The process of interpolating these sparse signals is scientifically complex, particularly when the spatiotemporal resolution of the measurement is low compared to the intrinsic scales of variability in the signal. Traditionally, statistical methods have been used to solve this problem. The most common procedure for ocean and climate sciences is *objective interpolation* [OI; Bretherton et al., 1976], alternatively called *kriging* or *Gaussian-process regression* in different fields. Essentially, this technique uses the spatiotemporal autocorrelation function to provide interpolation weights. Machine learning, however, offers an exciting range of new methods for interpolation which may provide more accuracy than classical methods [Bélisle et al., 2015, Zhou et al., 2017]. The LDEO group will apply ML to ocean remote-sensing interpolation and extrapolation problems, providing a use case to drive software and tutorial development.

The specific use case will be drawn from co-PI Abernathey's experience as part of the NASA SWOT Science Team, which will provide the next-generation of ocean sea level measurements. Current-generation Jason and other nadir altimeters measure sea-surface height (SSH) at the point just below their orbit paths. The upcoming Surface Water and Ocean Topography mission (SWOT) will instead measure narrow swaths of sea level. For both measurements, the process of moving from Level 2 to Level 4 data is fraught, due to both the distance spacing of the tracks

/ swaths (hundreds of km) and the slow repeat cycle of the mission (approx. 10 days for Jason, 20 days for SWOT); there is significant variability of sea level below these scales. The current state-of-the-art method uses a form of OI [Le Traon et al., 1998], which has been operationalized in the widely used Copernicus Ssalto/Duacs L4 products (formerly AVISO). New results suggest that deep convolutional neural networks (CNNs) could provide superior interpolation methods [Manucharyan et al., 2019]. The primary challenge in applying such methods to real Jason / SWOT data is the need to address unstructured / irregularly spaced observations, which CNNs cannot easily handle. We will research this problem, identify gaps in existing tools and ML architectures, and pursue software improvements in tandem with scientific research. The end result will be an experimental new L4 SSH product, plus open-source code to generate this product from the raw observations. These methods will likely also be applicable to other unstructured datasets, such as ARGO floats and surface drifters.

Beyond interpolation, oceanographers often wish to use remote sensing to infer derived quantities, far removed from the direct observations. For example, a long-standing problem is how to infer deep ocean velocities from surface observations, which lies more in the realm of extrapolation than interpolation [Wang et al., 2013]. After completion of the interpolation use case, we will turn our attention to this more difficult inference problem. This work will be conducted in close collaboration with the SWOT science team, with the first SWOT data anticipated to be released in 2021 or 2022. Emerging science applications from the Science Team will be used to guide further software development and demonstrations.

3.2 Land Model Parameterization Using ML-based Equation Discovery (Hamman)

Process-based hydrologic models play a central role in our ability to understand and simulate terrestrial systems. These models combine our best scientific understanding of basic physical processes (i.e. governing equations) with parameterizations that simulate processes that can not be explicitly formulated using first principles. The challenge scientists face is that many of the properties included in these model parameterizations (e.g. characteristics of vegetation and soil) are largely unknown. This challenge has motivated an area of research known as parameter estimation and rationalization. The goal is to find a set of "effective parameters" that translate observed information describing the system to parameter values the model can use. Fortunately, we now have a plethora of observations of the land surface, from a variety of Earth observing platforms (MODIS, Landsat, SMAP, ASTER, etc.) and in-situ measurements (USGS streamflow, FLUXNET, etc.), to develop data-driven models to robustly predict these "effective parameters".

Recent work on parameter estimation and regionalization demonstrated the potential of ML-based approaches for improving on the state of the art. For example, Samaniego et al. [2010] introduced the Multiscale Parameter Regionalization (MPR) technique, which uses calibrated pedotransfer functions (PTFs) to simultaneously incorporate spatial heterogeneity of physiographic characteristics while maintaining geographic and scale transferability of

parameters. While the MPR method was successful in demonstrating its the potential for the derivation of data-driven PTFs for hydrologic models, follow-on studies [e.g. Mizukami et al., 2017] have shown the method to be less performant in more complex models. Initial indications of why the MPR method does not seem to transfer well to more complex models seem to center around an over reliance on the fixed form of the the PTFs.

Our proposed application will explore the use of emerging techniques in machine learning to reformulate the MPR method. We specifically plan to test the applicability of two approaches for learning the PTFs: 1) equation discovery [i.e. Champion et al., 2019], and 2) inverse methods [i.e. Haber and Ruthotto, 2017]. We will test these approaches for parameterizing the Structure for Unifying Multiple Modeling Alternatives (SUMMA) hydrologic model [Clark et al., 2015]. Finally, we note that this application aligns well with ongoing land surface modeling activities at NCAR and that the SUMMA model is actively being integrated into NASA's Land Information System [LIS; Peters-Lidard et al., 2007] under separate support from the AIST program (80NSSC17K0541).

4 Management Plan

4.1 Roles and Coordination

Our team consists of data and Earth system scientists from the National Center for Atmospheric Research (NCAR), a federally funded research and development center and leader in supplying tools and technologies to the ESS community, Lamont-Doherty Earth Observatory at Columbia University, a leading research institute seeking fundamental knowledge about the natural world, Space Science and Engineering Center at the University of Wisconsin–Madison, a world leader in developing the algorithms and designing the ground and archive systems necessary to process atmospheric data collected from geostationary and polar-orbiting platforms, and Anaconda, a dynamic data-science startup company that is the leader in open-source scientific Python development. Each of these groups has demonstrated leadership and innovation in its respective discipline, along with the ability to build and maintain successful open-source software projects.

PI Hamman (NCAR) will provide overall project leadership and coordination. He will lead the development of the land-model equation-discovery science application (Section 3.2). He will also collaborate on the development of Xbatcher and continue contributing to the maintenance and development of the Xarray, Dask, and Intake software projects. Finally, as we describe below, he will continue to contribute to the NASA Earth Science Data Systems Working Groups.

Co-I Abernathey (Columbia) will lead the development of the ocean-interpolation science use case (Section 3.1), supervise the Ph.D. student, and provide technical leadership on the development of Xbatcher. He will also play a coordinating role among other development efforts in the Pangeo projects, including the Columbia's new Climate Data Science Lab.

Co-I Hoese (University of Wisconsin-Madison) will lead the technical development effort related to the Pytroll projects. He will serve as the coordination point between the Pangeo and Pytroll development teams as well as other software groups working with satellite instrument data like the GDAL, rasterio, and pyproj projects.

Co-Is Bednar and Augspurger (Anaconda Inc.) will lead the technical development of EarthML and coordinate changes and improvements to the Anaconda-maintained libraries used in the project (Dask, Dask-ML, Intake, HoloViz, etc.). They will also collaborate with the rest of the project team on the development of Xbatcher, while continuing to make substantive software contributions across the broader Pangeo ecosystem (such as Pandas, GeoPandas, Scikit-Learn, etc.) as needs are identified during this project.

Communication Detailed technical communication among team members will primarily be via the GitHub website, which is already the nexus of development for the Xarray, Dask, Jupyter, EarthML, Intake, HoloViz, Pytroll and Pangeo projects. GitHub offers powerful features for issue tracking, code merging, and continuous integration which have gained near universal adoption in the software development world. Informal discussions will be held in an online forum open to all Pangeo participants (currently https://gitter.im/pangeo-data/Lobby and https://discourse.pangeo.io/).

Finally, we will hold an annual in-person meeting in Austin, Texas in conjunction with the annual SciPy conference. The annual meeting will serve as a review of project progress, at which challenges can be addressed, unexpected problems corrected, and more detailed plans made for the coming year of work.

Community Engagement We recognize that the science applications described in section 3 represent a small fraction of the potential user community for Pangeo-ML. In response to this, we are actively working on developing connections with other ML groups. Our informal collaboration with the Radiant Earth Foundation (see letter of collaboration) as well as our continued involvement with the Pangeo ML Working Group are two examples of how we will work to disseminate the tools developed in this project to the broader community. Another key aspect of our community engagement plan is through the teaching of tutorials and short courses. Each of the investigators on this project regularly hosts workshops and tutorials at conferences (e.g. AGU, AMS, SciPy) and we will work to integrate new materials derived from this project into those activities.

Participation in NASA Earth Science Data Systems Working Groups (ESDSWG) 0.25 FTE has been budgeted so that PI Hamman will be able to continue his participation in the various ESDSWGs throughout the duration of this project. Dr. Hamman brings expertise in open-source software development, working with large geospatial datasets, and domain expertise in the climate and hydrologic sciences. He is also a member of the Pangeo steering council. Dr. Hamman is currently participating in the Cloud Analytics Reference Architecture and the Community Systems Integration working groups and, in the coming years, will seek to participate in new working groups that focus on machine-learning workflows that utilize NASA EO data.

Leveraging Past Efforts While the efforts described in this proposal are new and are not a direct continuation of previously awarded NASA funding, this project is designed to leverage previous efforts funded by the NSF-EarthCube, NASA-ROSES, NASA-ACCESS, and NASA-SBIR

Task	Description	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12
0	Support of Pangeo Software	X	X	X	X	X	X	X	X	X	X	X	X
1	**Pytroll Integration**												
1.1	Xarray Integration with Pytroll	X	X	X									
1.2	Dask Integration with Pytroll			X	X	X	X						
1.3	Intake Drivers for Satpy						X	X	X	X			
2	**Xbatcher Development**												
2.1	Add TensorFlow Dataset API to Xbatcher	X	X	X									
2.2	Add PyTorch Dataset API to Xbatcher			X	X								
2.3	Xbatcher-cache API					X	X	X	X				
3	**Demonstration and Application**												
3.1	Science application: Abernathey	X	X	X	X	X	X	X	X	X	X	X	X
3.2	Science application: Hamman	X	X	X	X	X	X	X	X	X	X	X	X
3.3	EarthML				X				X				X

Table 1: Proposed work plan outlining the anticipated key milestones for accomplishments.

programs. The scientific use cases will enhance data-analysis and data-management capabilities in ongoing NASA funding efforts including the SWOT science team (co-I Abernathey) and ongoing AIST projects at NCAR (PI-Hamman is contributing internally). NASA funding would enable major advancements to technologies that are already in heavy use by the Pangeo and NASA science communities.

Deploying Technologies to EOSDIS The technology built in this proposal is meant to work alongside current EOSDIS tools and enable science applications that take direct advantage of NASA data in the cloud. Because of the modular nature of the Pangeo ecosystem, individual technologies (e.g. Xarray, Dask, Intake) are readily adaptable to EOSDIS applications.

Utilizing Cloud Computing Resources We will be deploying the tools and science applications developed in this project on Amazon Web Services (AWS). Importantly, the tools in this proposal will be easily transferable to other commercial cloud computing options, such as Google Cloud and Microsoft Azure. Our budget includes a line item for cloud-computing resources to facilitate our project's development activities.

4.2 Timeline

As we described above, in section 2, our proposal includes four primary development activities: 1) improvement, maintenance, and support of high-value open-source technologies, 2) integration of data-manipulation libraries in the Pangeo and Pytroll projects, 3) development of software interfaces between Xarray and machine-learning libraries, and 4) development and dissemination of high-quality demonstrations and documentation.

Table 1 provides a quarter-by-quarter summary of the proposed project's timeline. In summary, all years will include improvement, maintenance, and support of the open-source software used and developed in this project. The Pytroll integration work will begin in year 1 and be completed early in Q1 of year 3. The Xbatcher development will begin in year 1 and complete by the end of year 2. Finally, the science applications and documentation efforts, include those with EarthML, will be spread over the project duration.

5 Open-Source Software Development Plan

5.1 Project Overview

The software used in this project is fully open source and is supported and co-developed by a community of developers at academic, industry, and philanthropic institutions. The Scientific Python ecosystem is also widely used in, and supported by, industry, particularly in the emerging fields of data science and machine learning [Muenchen, 2017]. We believe that contributing to and improving existing open-source libraries is the best way to serve the scientific programming community. Our team consists of maintainers and contributors from this community and we will continue to follow our strong track record in producing well-documented, well-tested, and heavily utilized open-source software.

Our goal is to make Earth-Observation–based machine-learning tasks easier for the scientific Python community. We will integrate data-manipulation software interfaces into the Pangeo and Pytroll projects, extend software interfaces between Xarray and machine learning libraries, and develop high-quality demonstrations and documentation for these new interfaces. Additionally we will provide improvements, maintenance, and support for the high-value open source technologies involved in these tasks.

5.2 Project Timeline

Task	Description	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12
0	**Support of Pangeo Software**	X	X	X	X	X	X	X	X	X	X	X	X
1	**Pytroll Integration**												
1.1	Xarray Integration with Pytroll	X	X	X									
1.2	Dask Integration with Pytroll			X	X	X	X						
1.3	Intake Drivers for Satpy						X	X	X	X			
2	**Xbatcher Development**												
2.1	Add TensorFlow Dataset API to Xbatcher	X	X	X									
2.2	Add PyTorch Dataset API to Xbatcher			X	X								
2.3	Xbatcher-cache API					X	X	X	X				
3	**Demonstration and Application**												
3.3	EarthML				X				X				X

Table 2: Proposed software work plan outlining the anticipated key milestones and accomplishments.

5.3 Open Sourcing Process History

License Information The packages involved in this proposal have the following OSS-compatible licenses: **Xarray** (Apache 2.0), **Dask**, **Dask-ML**, **HoloViz**, and **Jupyter** (BSD-3-Clause), **Satpy** (GPL-3.0), **Pyresample** (LGPL-3.0), **Xbatcher** (MIT), and **Intake** (BSD-2-Clause).

Code Repositories and Issue Tracking The GitHub website is used for code hosting, code review, and issue tracking for all software involved in this project. Each package has its own GitHub repository as shown in Table 3.

Project	Documentation	Code Repository and Issue Tracker
Xarray	http://xarray.pydata.org	https://github.com/pydata/xarray
Dask	https://dask.org	https://github.com/dask/dask
Dask-ML	https://ml.dask.org	https://github.com/dask/dask-ml
Jupyter	https://jupyter.org/	https://github.com/jupyter/notebook
Satpy	https://satpy.readthedocs.io	https://github.com/pytroll/satpy
Pyresample	https://pyresample.readthedocs.io	https://github.com/pytroll/pyresample
Xbatcher	https://xbatcher.readthedocs.io	https://github.com/rabernat/xbatcher
Intake	https://intake.readthedocs.io	https://github.com/intake/intake

Table 3: Documentation and code repository/issue trackers for the primary open source software projects to be developed as part of this project.

Documentation Documentation for all of these projects is automatically generated when changes are made, ensuring that users will see the most up-to-date information. Every package includes extensive API documentation and sections on basic usage, contributor guides, and installation instructions. Many of the projects include additional documentation on how the project is maintained and released. This focus on full documentation stems from our team's experience and long history of contributing to and maintaining open source software. The root page for each package's documentation can be found in Table 3.

5.4 Project Roles

Our team consists of key members of the scientific Python ecosystem and the packages involved in this proposal. We are confident in our ability to contribute to any Python library, but also play specific roles for certain packages. These roles are outlined below in Table 4.

5.5 Development and Testing Process Overview

Our team will follow standard open-source development processes adopted by many scientific Python libraries. This workflow allows for flexible code development, review, and parallel contributions. We will utilize the Git revision control system along with the GitHub collaboration website. Once a particular feature has been completed, bug fixed, or has its documentation changed, we will make GitHub "Pull Requests" (PRs) to begin a formal code review. GitHub's Pull Request feature has been configured for each project to perform automated testing and coding style checks. When all checks have passed and reviewers have approved the changes, the PR will be merged into the upstream public package source code. This updated source code will be released and DOIs made available via Zenodo.

Each software package involved in this project adopts a Continuous Integration (CI) approach to automated testing. This means that each project automatically runs tests for multiple platforms by using free CI services (e.g. Travis-CI, Appveyor, Microsoft Azure Pipelines). All contributions provided as GitHub Pull Requests are reviewed to verify that tests have been added for code paths affected.

Developer	Core Developer Role	Contributor Role
PI Hamman	Xarray, Zarr, Dask, Intake-Stac, Pangeo Cloud	Intake, Jupyter
Co-I Abernathy	Xarray, Zarr, Xbatcher, Pangeo Cloud	Intake, Dask
Co-I Hoese	Satpy, Pyresample	Dask, Xarray
Co-I Bednar	EarthML, HoloViz (Datashader, Panel, etc.)	Intake
Co-I Augspurger	Pandas, Dask, Dask-ML, Pangeo Cloud	Intake, Fsspec

Table 4: Core development and contributor roles for project PIs.

5.6 External Contribution Plan

Our team encourages external contributions in order to grow our community. External contributions for the mentioned packages are accepted as Pull Requests on GitHub and do not require a Contributor License Agreement (CLA). By following standard practices of the scientific Python community we hope to provide contributors a familiar and welcoming path for submitting changes to our software.

5.7 Communication Plan

GitHub's Issues feature will be the primary form of communication for feature requests, bug reports, and documentation changes. GitHub Pull Requests will be used for code review and discussing feature modifications.

5.8 Code of Conduct

The projects and communities involved in this proposal believe in providing a welcoming experience for all users, contributors, and other community members. The projects with larger communities have defined these community expectations in Code of Conduct documents available in each project's code repository. Many of these projects have adapted their Code of Conduct from Contributor Covenant (https://www.contributor-covenant.org/). Because most of our project activities will fall under the Pangeo umbrella, we will use the Pangeo governance documents, including its code of conduct (https://github.com/pangeo-data/governance/blob/master/conduct/code_of_conduct.md).

5.9 Security Considerations

The software involved in this project does not require user accounts to access nor does it collect usage telemetry (or other sensitive data). For these reasons, we do not state specific security considerations. We are however aware of the EOSDIS' Archiving, Distribution, and User Services Requirements Document (ADURD), and should our project venture into areas governed by this document, we will work with EOSDIS to ensure compliance.

References

Martín Abadi, Ashish Agarwal, and coauthors. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL http://tensorflow.org/. Software available from tensorflow.org.

Ryan Abernathey, Noah Brenowitz, Stephan Rasp, and Zhonghua Zheng. Pangeo-data ml-workflow-examples, Dec 2019. URL https://github.com/pangeo-data/ml-workflow-examples.

Eve Bélisle, Zi Huang, Sébastien Le Digabel, and Aïmen E Gheribi. Evaluation of machine learning interpolation techniques for prediction of physical properties. *Computational Materials Science*, 98:170–177, 2015.

Francis P Bretherton, Russ E Davis, and CB Fandry. A technique for objective analysis and design of oceanographic experiments applied to MODE-73. In *Deep Sea Research and Oceanographic Abstracts*, volume 23, pages 559–582. Elsevier, 1976.

Kathleen Champion, Bethany Lusch, J Nathan Kutz, and Steven L Brunton. Data-driven discovery of coordinates and governing equations. *Proceedings of the National Academy of Sciences*, 116 (45):22445–22451, 2019.

Martyn P Clark, Bart Nijssen, Jessica D Lundquist, Dmitri Kavetski, David E Rupp, Ross A Woods, Jim E Freer, Ethan D Gutmann, Andy W Wood, Levi D Brekke, et al. The structure for unifying multiple modeling alternatives (summa), version 1.0: Technical description. *NCAR Tech. Note NCAR/TN-5141STR*, 2015.

Dask Development Team. *Dask: Library for dynamic task scheduling*, 2016. URL http://dask.pydata.org.

Dask-ML Development Team. *Dask-ML: Scalable Machine Learning with Dask*, 2019. URL https://dask-ml.readthedocs.io/en/latest/.

Element 84. Cmr stac api proxy, Dec 2019. URL https://github.com/Element84/cmr-stac-api-proxy.

Eldad Haber and Lars Ruthotto. Stable architectures for deep neural networks. *CoRR*, abs/1705.03341, 2017. URL http://arxiv.org/abs/1705.03341.

Hamman. *Intake-stac: Intake interface to STAC data catalogs*, 2019. URL https://intake-stac.readthedocs.io/.

Joseph J Hamman. Pangeo, Nov 2019. URL https://medium.com/pangeo/the-pangeo-pattern-9a81ca4bad42.

Stephan Hoyer and Joseph J. Hamman. xarray: N-D labeled Arrays and Datasets in Python. *Journal of Open Research Software*, 2017. doi: 10.1109/mcse.2011.37.

PY Le Traon, F Nadal, and N Ducet. An improved mapping method of multisatellite altimeter data. *Journal of atmospheric and oceanic technology*, 15(2):522–534, 1998.

Georgy Manucharyan, Lia Siegelman, and Patrice KLEIN. State estimation of surface and deep flows from sparse ssh observations of geostrophic ocean turbulence using deep learning, Dec 2019. URL http://dx.doi.org/10.31223/osf.io/m8f3x.

Steven Miller and Debbie Hughes. The quant crunch: How the demand for data science skills is disrupting the job market. *Burning Glass Technologies*, 2017.

Naoki Mizukami, Martyn P Clark, Andrew J Newman, Andrew W Wood, Ethan D Gutmann, Bart Nijssen, Oldrich Rakovec, and Luis Samaniego. Towards seamless large-domain parameter estimation for hydrologic models. *Water Resources Research*, 53(9):8020–8040, 2017.

Robert A. Muenchen. The popularity of data science software, 2017. URL http://r4stats.com/articles/popularity/.

Christa D Peters-Lidard, Paul R Houser, Yudong Tian, Sujay V Kumar, James Geiger, S Olden, L Lighty, B Doty, P Dirmeyer, J Adams, et al. High-performance earth system modeling with nasa/gsfc's land information system. *Innovations in Systems and Software Engineering*, 3(3): 157–165, 2007.

Markus Reichstein, Gustau Camps-Valls, Bjorn Stevens, Martin Jung, Joachim Denzler, Nuno Carvalhais, et al. Deep learning and process understanding for data-driven earth system science. *Nature*, 566(7743):195–204, 2019.

Russ Rew and Glenn Davis. Netcdf: an interface for scientific data access. *IEEE computer graphics and applications*, 10(4):76–82, 1990.

Matthew Rocklin. Dask: Parallel computation with blocked algorithms and task scheduling. In Kathryn Huff and James Bergstra, editors, *Proceedings of the 14th Python in Science Conference*, pages 130 – 136, 2015.

Luis Samaniego, Rohini Kumar, and Sabine Attinger. Multiscale parameter regionalization of a grid-based hydrologic model at the mesoscale. *Water Resources Research*, 46(5), 2010.

Miguel-Angel Sicilia, Elena García-Barriocanal, and Salvador Sánchez-Alonso. Community curation in open dataset repositories: Insights from zenodo. *Procedia Computer Science*, 106:54 – 60, 2017. ISSN 1877-0509. doi: https://doi.org/10.1016/j.procs.2017.03.009. URL http://www.sciencedirect.com/science/article/pii/S1877050917302776. 13th International Conference on Current Research Information Systems, CRIS2016, Communicating and Measuring Research Responsibly: Profiling, Metrics, Impact,Interoperability.

Jinbo Wang, Glenn R Flierl, Joseph H LaCasce, Julie L McClean, and Amala Mahadevan. Reconstructing the ocean's interior from surface data. *Journal of Physical Oceanography*, 43 (8):1611–1626, 2013.

Wentian Zhou, Xin Li, and Daryl S Reynolds. Nonlinear image interpolation via deep neural network. In *2017 51st Asilomar Conference on Signals, Systems, and Computers*, pages 228–232. IEEE, 2017.

Data Management Plan

Our project will not produce new primary data; rather, it will analyze existing datasets that are already hosted in NASA cloud archives. Our primary research product will be software, both via contributions to existing open-source repositories (e.g. Xarray and Dask) and as new standalone packages (e.g. XBatcher). As we describe in our "Open Source Software Development Plan", the software will be hosted and version-controlled on the GitHub website, all code will be released with an open-source licenses, we will follow industry-standard best practices for software development, employing version control, continuous integration and testing, and comprehensive documentation and DOIs will be generated for all software releases via Zenodo [Sicilia et al., 2017].

Research artifacts and data products related to the science applications in our project will also be archived according to NASA's data management policies. Trained machine learning models will be archived on GitHub and Zenodo. Derived data products in excess of 50GB (Zenodo's stated maximum archive size) will be archived on cloud object storage (e.g. S3). We have included in our budget adequate time to achieve archiving of any such datasets.

Earth science data information policy: We are aware that NASA's policy is to "maximize access to data and to keep user costs as low as possible". This project will enhance access to data via tools that interface with NASA metadata repositories and follow best practices in cloud-computing design to avoid unnecessary costs associated with moving and downloading raw data.