



## TECHNICAL NOTE

# Toward a scalable framework for reproducible processing of volumetric, nanoscale neuroimaging datasets

Erik C. Johnson <sup>1,\*</sup>, Miller Wilt<sup>1</sup>, Luis M. Rodriguez<sup>1</sup>, Raphael Norman-Tenazas<sup>1</sup>, Corban Rivera<sup>1</sup>, Nathan Drenkow<sup>1</sup>, Dean Kleissas<sup>1</sup>, Theodore J. LaGrow<sup>2</sup>, Hannah P. Cowley<sup>1</sup>, Joseph Downs<sup>1</sup>, Jordan K. Matelsky<sup>1</sup>, Marisa J. Hughes<sup>1</sup>, Elizabeth P. Reilly<sup>1</sup>, Brock A. Wester<sup>1</sup>, Eva L. Dyer<sup>2,3</sup>, Konrad P. Kording<sup>4</sup> and William R. Gray-Roncal<sup>1,\*</sup>

<sup>1</sup>Research And Exploratory Development Department, Johns Hopkins University Applied Physics Laboratory, 11100 Johns Hopkins Rd., Laurel, MD, 20723 USA; <sup>2</sup>School of Electrical & Computer Engineering, Georgia Institute of Technology, 777 Atlantic Dr. NW, Atlanta, GA, 30332 USA; <sup>3</sup>Coulter Department of Biomedical Engineering, Georgia Institute of Technology, 313 Ferst Dr., Atlanta, GA, 30332 USA and <sup>4</sup>Department of Biomedical Engineering, University of Pennsylvania, 210 South 33rd St., Philadelphia, PA, 19104 USA

\*Correspondence address. Erik C. Johnson, Research And Exploratory Development Department, Johns Hopkins University Applied Physics Laboratory, 11100 Johns Hopkins Rd., Laurel, MD, 20723 USA. E-mail: [erik.c.johnson@jhuapl.edu](mailto:erik.c.johnson@jhuapl.edu)  <http://orcid.org/0000-0002-7397-8531>; William Gray-Roncal, Research And Exploratory Development Department, Johns Hopkins University Applied Physics Laboratory, 11100 Johns Hopkins Rd., Laurel, MD, 20723 USA. E-mail: [william.gray.roncal@jhuapl.edu](mailto:william.gray.roncal@jhuapl.edu)

## Abstract

**Background:** Emerging neuroimaging datasets (collected with imaging techniques such as electron microscopy, optical microscopy, or X-ray microtomography) describe the location and properties of neurons and their connections at unprecedented scale, promising new ways of understanding the brain. These modern imaging techniques used to interrogate the brain can quickly accumulate gigabytes to petabytes of structural brain imaging data. Unfortunately, many neuroscience laboratories lack the computational resources to work with datasets of this size: computer vision tools are often not portable or scalable, and there is considerable difficulty in reproducing results or extending methods. **Results:** We developed an ecosystem of neuroimaging data analysis pipelines that use open-source algorithms to create standardized modules and end-to-end optimized approaches. As exemplars we apply our tools to estimate synapse-level connectomes from electron microscopy data and cell distributions from X-ray microtomography data. To facilitate scientific discovery, we propose a generalized processing framework, which connects and extends existing open-source projects to provide large-scale data storage, reproducible algorithms, and workflow execution engines. **Conclusions:** Our accessible methods and pipelines demonstrate that approaches across multiple neuroimaging experiments can be standardized and applied to diverse datasets. The techniques developed are demonstrated on neuroimaging datasets but may be applied to similar problems in other domains.

Received: 20 April 2019; Revised: 19 August 2020; Accepted: 18 December 2020

© The Author(s) 2020. Published by Oxford University Press GigaScience. This is an Open Access article distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted reuse, distribution, and reproduction in any medium, provided the original work is properly cited.

**Keywords:** computational neuroscience; containers; electron microscopy; microtomography; optimization; reproducible science; workflows

## Introduction

Testing modern neuroscience hypotheses often requires robustly processing large datasets. Often the laboratories best suited for collecting such large, specialized datasets lack the capabilities to store and process the resulting images [1]. A diverse set of imaging modalities, including electron microscopy (EM) [1], array tomography [2], CLARITY [3], light microscopy [4], and X-ray microtomography (XRM) [5] will allow scientists unprecedented exploration of the structure of healthy and diseased brains. The resulting structural connectomes, cell type maps, and functional data have the potential to radically change our understanding of neurodegenerative disease.

Traditional techniques and pipelines developed and validated on smaller datasets may not easily transfer to datasets that are acquired by a different laboratory or that are too large to analyze on a single computer or with a single script. Prior machine vision pipelines for EM processing, for instance, have had considerable success [6–10]. However, these pipelines may require extensive configuration and are not scalable [8], may require proprietary software and have unknown hyperparameters [9], or are highly optimized for a single hardware platform [10].

In other domains, computer science solutions exist for improving algorithm portability and reproducibility, including containerization tools like Docker [11] and workflow specification such as the Common Workflow Language (CWL) [12]. Cloud computing frameworks enable the deployment of containerized tools [13,14], pipelines for scalable execution of Python code [15], and reproducible execution [16]. Workflow management and execution systems such as Apache Airflow [17] and related projects such as TOIL [18] and CWL-Airflow [19] allow execution of pipelines on scalable cloud resources. Despite the existence of these tools, a gap currently exists for extracting knowledge from neuroimaging datasets (due to the general lack of experience with these solutions, as well as a lack of neuroimaging-specific features). We propose a solution that includes a library of reproducible tools and pipelines, integration with compute and storage solutions, and tools to automate and optimize deployment over large (spatial) datasets. This gap is highlighted in Table 1 and discussed further in the Methods section; critically our proposed solution combines common workflow specifications, Dockerized tools, and automation for large-scale jobs over volumetric neuroimaging datasets.

We introduce a library of neuroimaging pipelines and tools, Scalable Analytics for Brain Exploration Research (SABER), to address the needs of the neuroimaging community. SABER introduces canonical pipelines for EM and XRM, specified in CWL, with a library of Dockerized tools. These tools are deployed using the workflow execution engine Apache Airflow [17] using Amazon Web Services (AWS) Batch to scale compute resources with imaging data stored in the volumetric database bossDB [20]. Metadata, parameters, and tabular results are logged using the neuroimaging database Datajoint [21]. Automated tools allow deployment of pipelines over blocks of spatial data, as well as end-to-end optimization of hyperparameters given labeled training data.

We demonstrate the use of SABER for 3 use cases critical to neuroimaging using EM, XRM, and light microscopy methods as exemplars. While light microscopy is commonly used to image cell bodies and functional activity with calcium markers, EM of-

fers unique insight into nanoscale connectivity [22–25], and XRM allows for rapid assessment of cells and blood vessels at scale [5,26, 27]. These approaches provide complementary information and have been successfully used on the same biological sample [5] because XRM is non-destructive and compatible with EM sample preparations and light microscopy preparations. Being able to extract knowledge from large-scale volumes is a critical capability, and the ability to reliably and automatically apply tools across these large datasets will enable the testing of exciting new hypotheses.

Our integrated framework is an advance toward easily and rapidly processing large-scale data, both locally and in the cloud. Processing these datasets is currently the major bottleneck in making new, large-scale maps of the brain—maps that promise insights into how our brains function and are affected by disease.

## Findings

### Pipelines and tools for neuroimaging data

To address the needs of the neuroimaging community, we have developed a library of containerized tools and canonical workflows for reproducible, scalable discovery. Key features required for neuroimaging applications include:

- Canonical neuroimaging workflows specified in CWL [12] and containerized, open-source image processing tools
- Integration of workflows with infrastructure to deploy jobs and store imaging data at scale
- Tools to optimize workflow hyperparameters and automate deployment of imaging workflows over blocks of data

Building on existing tools, our framework provides a more accessible approach for neuroimaging analysis and can enable a set of use cases for the neuroscientist by improving reproducibility. Details on adoption can be found in the Section “Required background and getting started.”

To ensure broad impact, the SABER is designed to be as generalizable as possible. The core abilities to schedule and launch Dockerized workflows are applicable to a wide range of volumetric datasets provided that (i) Dockerized tools exist, (ii) CWL workflows can be specified, and (iii) raw data can be accessed from existing volumetric repositories [20,28, 29], local files, or cloud buckets. The standardized workflows described below are developed specifically for EM and XRM. These workflows perform generalized, repeated processing techniques such as classification, object detection, and 2D and 3D segmentation, but with parameters and weights specific to these modalities. Users may be able to adapt these tools to additional problems with the use of annotated training data and appropriate tuning.

### Standardized workflows and tools

While many algorithms and workflows exist to process neuroimaging datasets, these tools are frequently laboratory and task specific. As a result, teams often duplicate common infrastructure code (e.g., data download or contrast enhancement) and re-implement algorithms, when it would be faster and more reliable to instead reuse previously vetted tools. This hinders

**Table 1:** Comparison of existing projects related to workflow execution of neuroimaging pipelines

Feature	SABER	CWL-Airflow	TOIL	Galaxy	Air-tasks	Kubernetes
Purpose	Workflow management system	Workflow management system	Workflow management system	Workflow management system	Workflow management system	Distributed container orchestration
Container support	Yes	Yes	Yes	Yes	Yes	Yes
Workflow description	CWL	CWL	CWL or WDL	Custom (CWL beta)	Custom Python	NA
Computational background	Novice-expert	Novice-expert	Novice-expert	Novice-expert	Intermediate-expert	Expert
Installation	docker-compose	pip install	pip install	Install scripts	docker-compose	Cluster configuration
Cloud support	AWS	Planned	Multiple cloud providers	Multiple cloud providers	Docker Infrakit	Multiple cloud providers
Volumetric database	bossDB, DVID	None	None	None	cloud-volume	NA
Parallel processing model	Block-merge	None	None	None	Block-merge	NA
Workflow deployment for neuroimaging	Yes	No	No	No	Yes	NA
Workflow optimization for neuroimaging	Yes	No	No	No	No	NA
Tool benchmarking and datasets	Yes	No	No	No	No	NA
EM tool library	Yes	No	No	No	Yes	NA
Tools for other modalities	Yes	No	No	No	No	NA

SABER delivers integrated containerized tools, a standardized workflow and tool description, and a volumetric database. It also provides tools for automating deployment over datasets by dividing into blocks (block-merge) and optimization of workflows. The most comparable tools are other workflow management systems such as CWL-Airflow, TOIL, Galaxy, and Air-tasks. Air-tasks provides similar capabilities but lacks support for common workflow descriptions and tool optimization and offers less flexibility for users. Similar projects such as TOIL, Galaxy, and CWL-Airflow lack neuroimaging-specific features to enable the use cases described in Methods. Scalable cluster systems, such as Kubernetes, provide essential functionality to deploy containers at scale but need capabilities built to manage workflows and data movement and are complementary to workflow management systems such as SABER. The SABER project adds critical features for neuroimaging by (i) interfacing with existing solutions; (ii) providing a library of portable, Dockerized neuroimaging tools; and (iii) providing scripting to analyze large-scale neuroimaging datasets. NA: not applicable; WDL: Workflow Definition Language.

attempts to reproduce results and accurately benchmark new image-processing algorithms.

In our framework, workflows are specified by CWL pipeline specifications. Individual tools are then specified by an additional CWL file, a container file, and corresponding source code. This ensures a modular design for pipelines and provides a library of tools for the neuroscientist. This library of pre-packaged tools and workflows helps reduce the number of computational frameworks and software libraries that users need to be familiar with, helping to limit the computational experience required to run these pipelines.

Initially, we have implemented 2 canonical pipelines for EM and XRM processing. For EM, we estimate graphs of connectivity between neurons from stacks of raw images. Given XRM images, we estimate cell body position and blood vessel position. Each of these workflows is broken into a sequence of canonical steps. Such a step-wise workflow can be viewed as a directed acyclic graph (DAG). Each step of a pipeline is implemented by a particular containerized software tool. The specific tools implemented in our reference canonical pipelines are discussed below.

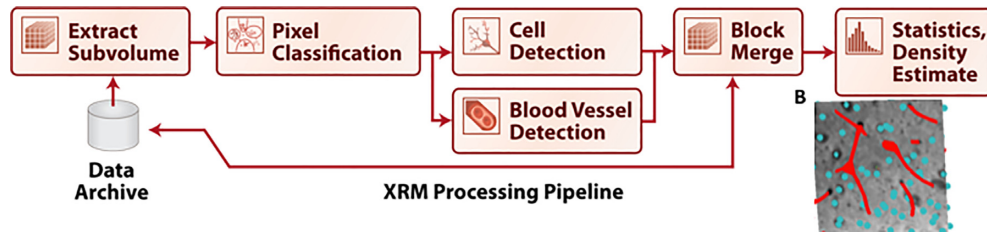
#### Cell detection from X-ray microtomography and light microscopy

XRM provides a rapid approach for producing large-scale sub-micron images of intact brain volumes, and computational workflows have been developed to extract cell body densities and vasculature [5]. Individual XRM processing tools have been developed for tomographic reconstruction [30], pixel classification

[31], segmentation of cells and blood vessels [5], estimation of cell size [5], and computation of the density of cells and blood vessels [5]. Running this workflow on a volume of X-ray images produces an estimate of the spatially varying density of cells and vessels. Samples of 1 m<sup>3</sup> size (100 GB) can be imaged, reconstructed, and analyzed in a few hours [5].

To implement a canonical XRM workflow, we define a set of steps: extracting subvolumes of data, classifying cell and vessel pixel probabilities, identifying cell objects and vasculature, merging the results, and estimating densities. Details on data storage and access can be found in the Framework Components section. We defined Dockerized tools implementing a random forest classifier, a Gaussian mixture model, and a U-net [32] for pixel classification and the cell detection and vessel detection strategies [5]. These tools provide a standard reference for the XRM community, and modular replacements can be made as new tools are developed and benchmarked against this existing standard. Fig. 1 shows this canonical workflow for XRM data, with each block representing a separate containerized tool. Also shown in Fig. 1B is example output from running the pipeline, highlighting the resulting cell body positions and blood vessels.

These same tools can also be applied (with appropriate re-training) to detecting cell bodies from light microscopy data, such as from the Allen Institute Brain Atlas [4]. Here the same pipeline tools can be reused to detect cell bodies using the step for pixel classification followed by the step for cell detection. This result demonstrates the application of these tools across modalities and datasets to ease the path to discover.



**Figure 1:** Workflow for processing XRM data to produce cell and vessel location estimates. Raw pixels are used to predict probabilities of boundaries, followed by detection of cell bodies and blood vessels. Finally, cell density estimates are created. A, reconstruction pipeline; B, reconstruction of the detected cells and blood vessels in the test volume. Cells are shown as spheres and blood vessels as red lines.

### Deriving synapse-level connectomes from electron microscopy

Several workflows exist to produce graphs of brain connectivity from EM data [6,7,10], including an approach that optimizes each stage in the processing pipeline based on end-to-end performance [8]. However, these tools were not standardized into a reproducible processing environment, making reproduction of results and comparison of new algorithms challenging.

We have defined a series of standard steps required to produce brain graphs from EM images, seen in Fig. 2. First, data are divided into subvolumes; cell membranes are estimated for each volume. Next, synapses are estimated and individual neurons are segmented from the data. After this, synaptic connections must be associated with neurons, and results merged together across blocks. Then a graph can be generated by iterating over each synapse to find the neurons representing each connection. Many tools have been developed for various sections of this pipeline, and a single tool may accomplish multiple steps of the pipeline. Examples of tools for membrane segmentation include convolutional neural network [33] and U-net [32] approaches. Synapse detection has been achieved using deep learning techniques and random forest classifiers [34,35]. Neural segmentation has been previously done using agglomeration-based approaches [36] and automated selection of neural networks [9]. For our initial implementation of this workflow, we create CWL specifications and containerized versions of U-nets [32] for synapse and membrane detection and use the GALA tool [37] for neuron segmentation, and algorithms for associating synapses to neurons and generating connectomes [8].

When creating this canonical pipeline for EM processing, our initial implementation goal is not to focus on pipeline performance in the context of reconstruction metrics. Rather, we aim to provide a reference pipeline for scientists and algorithm developers. For scientists, this provides an established and tested pipeline for initial discovery. For algorithm developers, this pipeline can be used to benchmark algorithms that encompass  $\geq 1$  step in the pipeline.

### Optimization and deployment of workflows

To process modern neuroimaging datasets, users need more than standardized pipelines and the ability to deploy them to individual blocks of data. Scaling these workflows to current datasets requires specialized interfaces to distribute jobs over large volume and tune them to new data. The SABER project provides (i) a parameterization API to distribute jobs over large volumes of data and (ii) an optimization API to train pipelines and fine-tune hyperparameters for new datasets.

To apply SABER workflows to large volumetric datasets, such as those hosted in bossDB [20], a parameterization API allows control over creating blocks from large datasets (by specifying sizes and overlap of blocks in each dimension), running

pipelines on each block, and merging results (i.e., a distribute-collect approach). A second parameter file specifies these desired parameters and can be used with any compatible workflow to deploy it to a new dataset. Deployment scripts enable rapid configuration and deployment of workflows for new datasets.

To tune SABER workflows for new datasets, it is necessary to train the parameters of the pipeline, including any hyperparameter optimization (Fig. 3). Our tools currently require a small volume of labeled training data from the new dataset (although recent efforts are also exploring unsupervised methods [38]). To perform the hyperparameter search, we pursue an optimization strategy that assumes a black-box workflow, avoiding assumptions such as differentiability of the objective function. SABER makes it possible to iteratively select parameters, schedule parallel jobs, and collect results. This approach supports both batch and sequential optimization approaches. Initially, we implemented a simple grid search, random search, and the adaptive search method shown in Fig. 3, based on random resampling [39]. This will be expanded to techniques such as sequential Bayesian optimization [40] and convex bounding approaches [41] to develop a library of readily available, proven techniques. To provide benchmarking for these approaches, the team hosts available ground truth data (e.g., [23] for EM), and scoring tools to compute metrics such as precision-recall or f1-score.

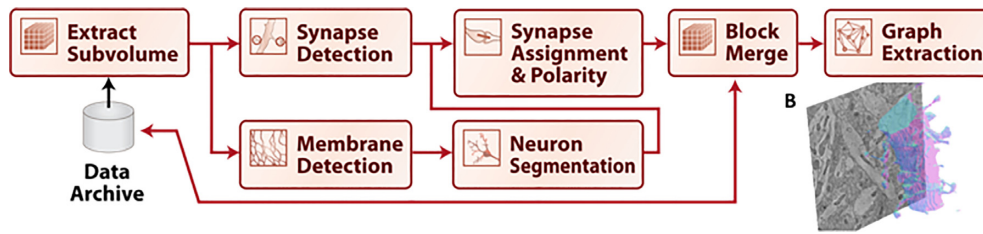
### Datasets for benchmarking workflows

A critical feature for new users, as well as developers of new containerized tools, is the availability of benchmark datasets for deriving synapse-level connectomes from EM, as well as segmentation of cell bodies and vasculature from XRM data. Datasets are hosted in the bossDB system [20, 42] for this purpose. For testing XRM pipelines, data from the datasets “Dyer et al. 2017” [5] and “Prasad et al. 2020” [43] can be used. These datasets contain different brain regions including labels of cell bodies and vasculature for training new users and developing new algorithms. Similarly, for EM data, datasets such as “Kasthuri et al. 2015” [23] provide EM data along with segmentation and synapse labels. These similarly enable new users and algorithm developers to compare to existing data and approaches.

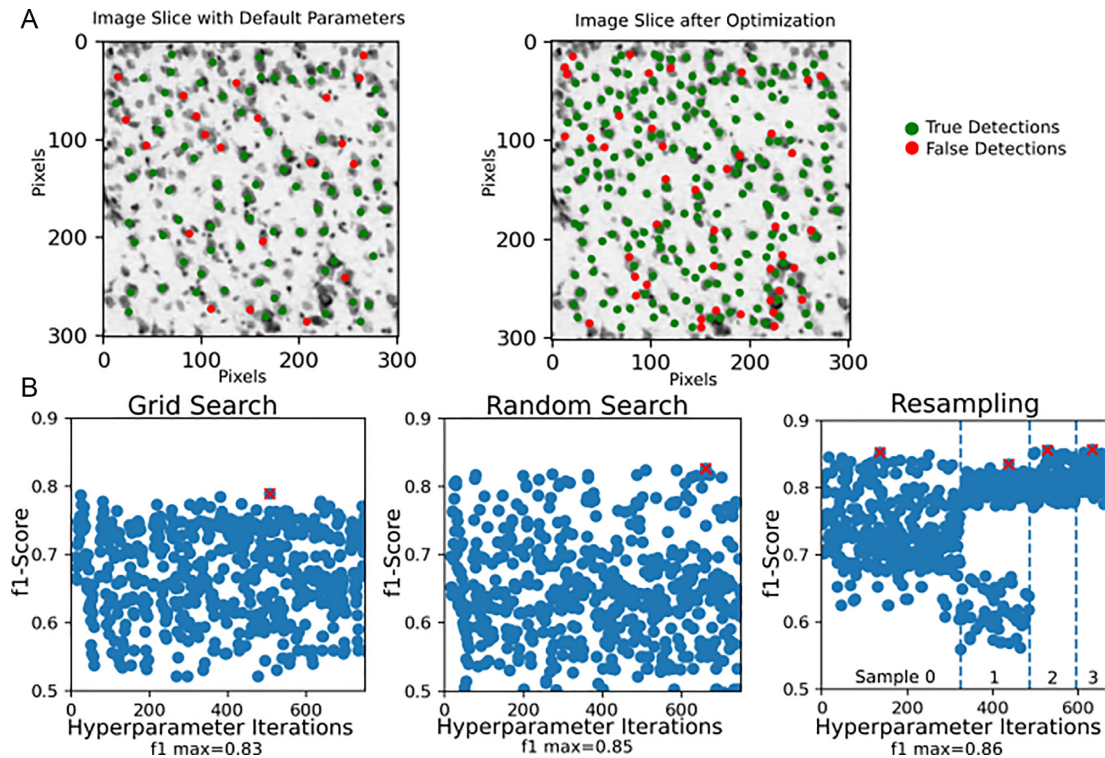
## Neuroimaging Use Cases

### Use case 1: Pipeline optimization

When a new neuroimaging dataset is being collected, it is often necessary to fine-tune or retrain existing pipelines. This is typically done by labeling a small amount of training data, which can often be labor intensive, followed by optimizing the automated image-processing pipeline for the new dataset. These pipelines



**Figure 2:** Canonical workflow for graph estimation in EM data volumes. This workflow provides the ability to reconstruct a nanoscale map of brain circuitry at the single-synapse level. The procedure of mapping raw image stacks to graphs representing synapse-level connectomes consists of synapse and membrane detection, segmentation of neurons, assignment of synapses, merging, and graph estimation. A, reconstruction pipeline; B, example segmentation of a neuron from a block of data.



**Figure 3:** Use case of optimizing a pipeline for light microscopy data, comparing grid search, random search, and the random resampling approach described in the text. We demonstrate these tools on a light microscopy dataset, leveraging methods originally developed for XRM—showcasing the potential for applying tools across diverse datasets. The framework allows a user to easily compare the trade-offs of different approaches for a particular dataset. The maximum f1 score for each approach is marked with a red cross. Automating this process using SABER allows for rapid deployment and optimization.

consist of heterogeneous tools with many hyperparameters and are not necessarily end-to-end differentiable.

Users can execute the optimization routines using a simple configuration file to specify algorithms, parameter ranges, and metrics. Fig. 3 demonstrates the application of 3 algorithms for pipeline optimization. We choose the Allen Institute for Brain Science Reference Atlas [4] as a demonstration of generalization beyond EM and XRM datasets. To optimize the pipeline, this example optimizes over the following parameters: the initial threshold applied to the probability map, size of circular template, size of circular window used when removing a cell from the probability map, and the stopping criterion for maximum correlation within the image. The user specifies the range of each parameter.

Our framework supports implementations of different optimization routines, such as random selection of parameters with resampling, as seen in Fig. 3. Random selection of parameters

often produces results comparable to those of grid search, and users may need to explore algorithms to find an approach that works well for the structure of their pipeline [39]. For the resampling approach, we initially choose parameters at random and then refine search parameters by choosing new parameters near the best initial set, with the user setting a maximum number of iterations. Fig. 3B shows a parameter reduction of 20% at each resampling, leading to a more efficient parameter search and improved performance. Using SABER, it is possible for a user to explore the trade-offs for a range of hyperparameter optimization routines.

### Use case 2: Scalable pipeline deployment

The second critical use case of interest to neuroscientists is the deployment of pipelines to large datasets of varying sizes. Datasets may be on the order of gigabytes or terabytes, as in

XRM, to multiple petabytes, as in large EM volumes used for connectome estimation. SABER provides a framework for blocking large datasets, executing optimized pipelines on each block, then merging the results through a functional API. Given a dataset in a volumetric database, such as bossDB, our Python scripts control blocking, execution, and merging. Results are placed back into a database for further analysis, or stored locally. An example of this use case for XRM data can be seen in Fig. 4, and another example of this use case for extracting synapse-level connectomes can be seen in Fig. 5.

### Use case 3: Benchmarking neuroimaging algorithms

The third major use case applies to developers implementing new algorithms for neuroimaging datasets. Owing to tools being written in a variety of languages for a variety of platforms, it has been difficult for the community to standardize comparison between algorithms. Moreover, it is important to assess end-to-end performance of new tools in a pipeline that has been properly optimized. Without this comparison, it is difficult to directly compare algorithms or their impact. Using the specified pipelines, a new tool may subsume 1 or more of these steps, with the specification defining the inputs and outputs. A new CWL pipeline can be quickly specified with the new tool replacing the appropriate step or steps. Hyperparameter optimization can be run on each example to compare tools, leveraging reference images and annotations for the pipelines provided in SABER.

## Discussion

We have developed a framework for neural data analysis along with corresponding infrastructure tools to allow scalable computing and storage. We facilitate the sharing of workflows by compactly and completely describing the associated set of tools and linkages. Future enhancements will introduce versioning to track changes in workflows and tools.

The SABER project aims to support multiple modalities, focusing initially on EM and XRM data through the development of containerized tools for different steps such as synapse and cell detection. The same tools can be used for different steps of both workflows. For instance, our U-net [32] tool can be used to generate probability maps for synapses, cell bodies, or cell membranes when training with different data. The framework also allows for joint analysis of co-registered datasets using our CWL pipelines using different parameterized sweeps. The user can then use simple Python scripts to pull and analyze any parts of these data.

While the SABER project has focused on tools for processing large EM and XRM datasets, many of the tools and infrastructure developed would also be of interest to researchers investigating light microscopy, positron emission tomography, and fMRI. The features of SABER are most appropriate for large-scale volumetric data, where records are large (gigabytes or larger) and it is difficult to process a dataset in memory. Therefore, larger light microscopy datasets may benefit the most from SABER. The developed tools focus on canonical problems such as object detection, 2D segmentation, and 3D segmentation. These are generally useful for structural neuroimaging datasets and may be reused in other contexts.

Our goal is to establish accessible reference workflows and tools that can be used for benchmarking new algorithms and assessing performance on new datasets. Moving forward, we will encourage algorithm developers to containerize their solutions for pipeline deployment and to incorporate state-of-the-

art methods. Through community engagement, we hope to grow the library of available algorithms and demonstrate large-scale pipelines that have been vetted on different datasets. We also hope to recruit researchers from different domains to explore how these tools apply outside of the neuroimaging community.

Prior solutions have taken different approaches to processing neuroimaging data. For example, the workflow execution engine LONI has been used for processing EM data [8] but requires extensive configuration and is not scalable to very large volumes. The SegEM framework [9] offers extensive features for optimizing and deploying EM pipelines but is specifically focused on neuron segmentation from EM data and is tied to a MATLAB cluster implementation. Highly optimized pipelines can be deployed on a single workstation [10], which is ideal for proven pipelines as part of ongoing data collection but is limited in developing and benchmarking new pipelines.

A major strength of the SABER approach is the use of CWL to provide a common specification for workflows, which has considerable advantages compared to workflow managers with specific Python syntax (e.g., [15, 45]). The common, interoperable standard is important to allow reuse of the SABER workflows in other workflow managers as they continue to evolve. This approach also encourages tools developed for other open-source projects to be deployed using the SABER system.

A limitation of our existing tooling is interactive visualization. Although we provide basic capabilities, additional work is needed to interrogate raw and derived data products and identify failure modes. We are extending the open-source packages substrate [46] and neuroglancer [44] to easily visualize data inputs and outputs of our workflows and tools.

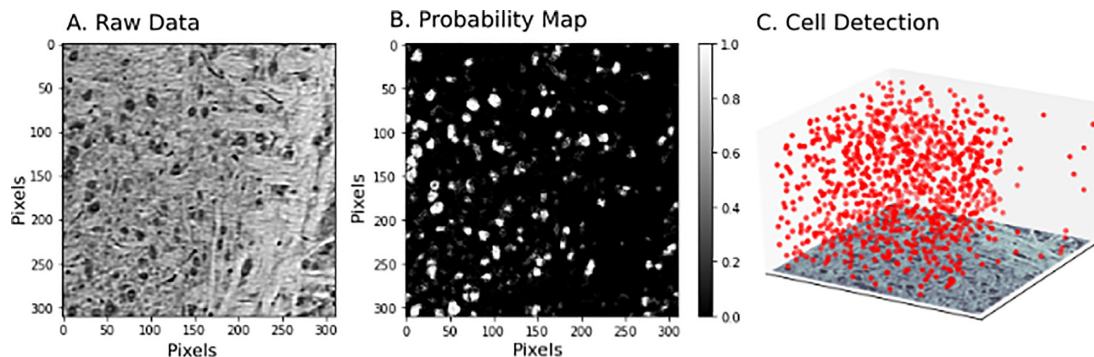
Scalable solutions for containers such as Kubernetes [13] and general workflow execution systems like Apache Airflow [17] have provided the ability to orchestrate execution of containers at scale. These solutions, however, lack workflow definitions, imaging databases, and deployment tools to enable neuroimaging use cases. SABER builds on top of these technologies to enable neuroimaging use cases while avoiding the specialized, one-off approaches often used in conventional neuroimaging pipelines.

Our solution leverages many powerful existing third-party solutions (e.g., AWS, Apache Airflow). While this allows use of powerful modern software packages and shared development, it creates a risk if these technologies are not supported and developed in the future. While it is not possible to completely mitigate this risk, the modular strategies for storage and computation, described below, help to mitigate this challenge by allowing components related to these services to be replaced. The key dependency is Apache Airflow, but even in this case the workflows and Dockerized tools have potential applications with future workflow managers.

## Potential Implications

While our initial workflows focus on XRM and EM datasets, many of these methods can be easily deployed to other modalities such as light microscopy [47], and the overall framework is appropriate for problems in many domains. These include other scientific data analysis tasks as varied as machine learning for processing noninvasive medical imaging data or statistical analysis of population data.

Code, demonstrations, and results of the SABER platform are available on GitHub under an open-source license, along with documentation and tutorials (see below). We make SABER avail-



**Figure 4:** Example deployment of pipeline over spatial dataset, in this case cell detection in XRM data. An example slice of raw data can be seen in A. The pipeline in Fig. 1 was used to classify pixels (B) and detect cells. From the cells, a 3D scatter plot of the positions of the cell centers was generated (C).

able to the public with the expectation that it will help to enable and democratize scientific discovery of large, high-value datasets and that these results will offer insight into neurally inspired computation, the processes underlying disease, and paths to effective treatment. Contributors and developers are also encouraged to visit and join the open-source developers on the project.

Future work will focus on usability, while integrating SABER into existing open source frameworks for data storage and visualization (e.g., [20], [44]). In an effort to lower the barriers for new users, this work will include GUIs, as well as the development of additional reference pipelines. Integration with datastores like bossDB will enable a common ecosystem for new users to find storage, processing, and visualization in a common location.

## Methods

### Existing software solutions

For small-scale problems, individual software tools and pipelines that are fully portable and reproducible have been produced (e.g., [48]), but this challenge has not yet been solved at the scale of modern EM and XRM volumes.

Many tools have become available for scalable computation and storage, such as Kubernetes [13] and Hadoop [49], which enable the infrastructure needed for running containerized code at scale. However, such projects are domain agnostic and do not necessarily provide the features or customization needed by a neuroscientist. As scalable computation ecosystems, these solutions can be integrated as the back end for workflow management systems such as SABER.

Traditional workflow environments (e.g., LONI Pipeline [50], Nipype [45], Galaxy [51], and Knime [52]) provide a tool repository and workflow manager but require connection to a shared compute cluster to scale. All of these systems rely on software that is installed locally on the cluster or local workstation, and can result in challenging or conflicting configurations that slow adoption and hurt reproducibility.

New frameworks for workflow execution have been developed but solve only a subset of the challenges for neuroimaging. Boutiques [53] manages and executes single, command line executable neuroscience tools in containers. Pipelines must be encapsulated in a single tool, meaning that coding is required to swap pipeline components. Dray [54] executes container-

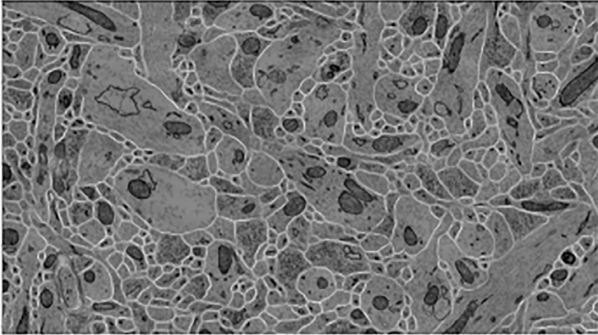
based pipelines as defined in a workflow script. While Dray contains some of the core functionality to execute container-based pipelines, non-programmers cannot easily use the system and it is limited in the types of workflows that are supported.

Similarly, Pachyderm [14] offers execution of containerized workflows but lacks support for storage solutions appropriate for neuroimaging, as well as optimization tools needed for these neuroimaging pipelines. Workflow execution engines such as TOIL [18] and CWL-Airflow [19] are closely related to SABER, providing lightweight Python solutions for workflow scheduling. However, like Pachyderm, they lack the automation tools and storage scripts required by neuroimaging applications. The most closely related tool is Air-tasks [55], which provides tools to automate deployment of neuroimaging pipelines. Air-tasks, however, provides fewer capabilities to the user and does not support a common workflow specification or explicitly support optimization or benchmarking.

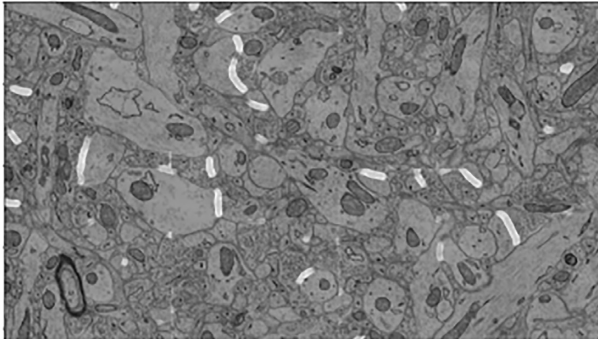
Table 1 breaks down this comparison between SABER and existing workflow managers and execution solutions for scientific computing. In general, neuroimaging applications benefit from several key features that are not provided in these more general-purpose scientific workflow approaches owing to the use of volumetric data, few large datasets (versus many smaller images in a large collection), and the need for tool cross-compatibility. SABER delivers these key features through the use of standardized workflows, containerized tools, automation of deployment over volumetric data (as opposed to processing individual records), and the ability to optimize pipelines. The closest existing solutions are workflow managers such as TOIL [18], Galaxy [51], and CWL-Airflow [19]. These approaches are powerful but focused on other problems in bioinformatics, such as gene sequence analysis, consisting of many small records. SABER adds the necessary features to provide these capabilities for the neuroimaging community.

While existing pipeline tools like LONI [50] and Nipype [45] enable the execution of scientific workflows, they still lack a few key features for the neuroimaging user and may limit the portability and utility of workflows. SABER provides a library of tools required for modern segmentation and detection problems on EM and XRM data, including GPU-enabled deep neural network tools. These tools, and their corresponding CWL definitions, are useful in any system that can support them, rather than being specific to a workflow manager, as with LONI and Nipype. We enable the use and sharing of Dockerized tools and standardized workflows within and beyond the SABER framework.

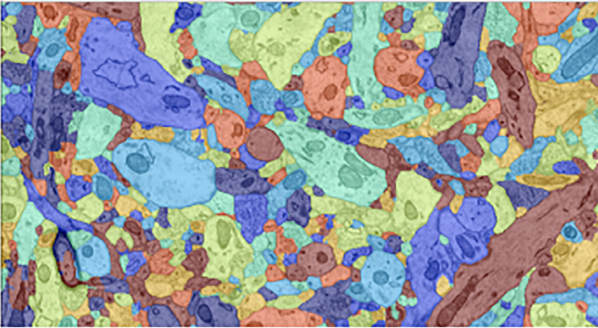
## A. Cell Membrane Detection



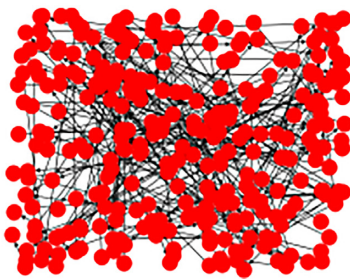
## B. Synapse Detection



## C. Cell Segmentation



## D. Graph Generation



**Figure 5:** Example deployment of EM segmentation pipeline to extract graphical models of connectivity from raw images. The processing pipeline (Fig. 1) consists of neural network tools to perform (A) membrane detection and (B) synapse detection. This is followed by a segmentation tool (C). Finally, segmentation and synapses are associated to create a graphical model. Visualizations of segmentations are done with Neuroglancer [44], a tool compatible with SABER and integrated with the bossDB [20] system.

## SABER

To overcome limitations in existing solutions, SABER provides canonical neuroimaging workflows specified in a standard workflow language (CWL); integration with a workflow execution engine (Airflow), imaging database (bossDB), and parameter database (Datajoint) to deploy workflows at scale; and tools to automate deployment and optimization of neuroimaging pipelines. Our automation tools include end-to-end hyperparameter optimization methods and deployment by dividing data into blocks, executing pipelines, and merging results (block-merge). In our repository, this is broken into 2 key components. The first is CONDUIT, which is the core framework for deploying workflows. The second is SABER, which contains the code, Dockerfiles, and CWL files for the workflows (Fig. 6). A comparison of SABER/CONDUIT to existing solutions is seen in Table 1.

The core framework (called CONDUIT) is provided in a Docker container to reduce installation constraints and increase portability (Fig. 6). The core framework interfaces with scalable cloud compute and storage resources, as well as local resources. The user interacts via command line tools and can visualize the status of workflows using Airflow’s GUI. Each tool used in the workflows will also be built into a separate image.

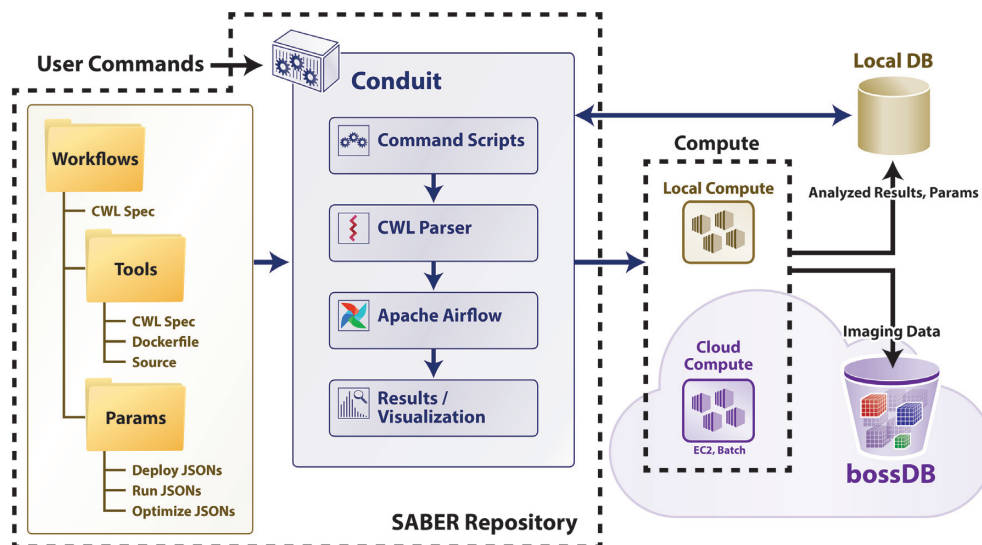
In our CONDUIT framework (Fig. 6 highlights the architecture of the system), workflows and tools are defined with CWL v1.0 specifications. Tools additionally include Dockerfiles and source code. Parameter files contain user-specified parameters for optimization and deployment of pipelines. The features of CONDUIT include parsing the CWL parameters and deploying workflows, as in the CWL-Airflow project [19]. Features added on top of the existing CWL-Airflow functionality include an API for parameterizing jobs for deployment over chunks of data in large volumetric datasets (specified by coordinates), iterative execution of the same workflow with different parameters (for parameter optimization), and logging of metadata and job results. Moreover, wrappers allow for the use of local files and cloud files (S3) for intermediate results with the same workflows and minimal re-configuration.

The GitHub repository [56] contains both our CONDUIT framework and the SABER workflows and tools, as visualized in Fig. 6. The CONDUIT framework consists of the Python code and scripts that build upon CWL and Airflow to enable the deployment of workflows. The SABER workflow code contains the tools, Dockerfiles, CWL definitions for tools, CWL definitions for workflows, and example job files. This structure emphasizes the portability of SABER tools—the use of Docker and CWL encourages their reuse in other contexts where the full power of the framework may not be needed (e.g., running on small, locally stored datasets).

## Framework components

The overall structure of SABER is seen in Fig. 6 and consists of tools, workflows, parsers for user commands, workflow execution, and cloud computation and storage. Workflows, found in the SABER component, consist of code, Dockerfiles, and CWL files. The core functionality of parsing workflows, running Airflow, and scheduling jobs is found in the CONDUIT component.





**Figure 6:** The architecture and components of SABER. Tools, workflows, and parameters for individual use cases (optimization, deployment) are captured in a file structure using standardized CWL specifications and configuration files. The core of the framework (called CONDUIT) is run locally in a Docker container. CONDUIT consists of scripts to orchestrate deployment and optimization, a custom CWL parser, Apache Airflow for workflow execution, and tools to collect and visualize results. Containerized tools are executed locally or using AWS Batch for a scalable solution. The bossDB provides a solution for scalable storage of imaging data, and a local database is used for storing parameters and derived information. JSON: JavaScript Object Notation.

#### SABER workflow library

The SABER subproject consists of a library of code, tools, and workflows. Each SABER tool must have a corresponding Dockerfile. Tools and workflows are specified following CWL specifications. To package a tool for SABER, a developer must

- Provide a Dockerfile for the tool
- Use command line arguments to specify file input and outputs (which can be read as any local file the tool can use)
- Provide a CWL tool file with tool parameters and input and output file names specified

Optionally, developers can choose to print metrics, scores, or other information on the command line. When building workflows, tools are wrapped to allow for either local or cloud execution and no additional requirements are placed on the tool developer.

Workflows are specified using standard CWL syntax. To specify local versus cloud execution, the CWL “doc” flag can be set to run with completely local compute and storage. Individual step “hints” can be used to specify that an individual step should use local compute resources. GPU resources can be used through configuration of the system Docker installation. Workflow parameters are also specified with standard CWL files.

To enable our neuroimaging use cases, parameter sweeps are specified with a new custom parameterization file. This specifies the parameter start, stop, step, and overlap. A typical use case is the specification of boundaries of a large volumetric dataset (xmin, xmax, ymin, ymax, zmin, zmax, and stepsize). Any parameter specified by a tool CWL can be included in the parameterization file.

To enable hyperparameter optimization of pipelines, a similar format to parameterization is used to specify which parameters are to be optimized and the range of these parameters, as well as the algorithm (e.g., grid or random search). A CWL hint is added to the workflow indicating the name of the optimization metric for each step, which will be parsed from standard out. This allows the specification of multiple objective functions or metrics for each workflow stage.

#### CONDUIT Docker container

The CONDUIT component (Fig. 6) contains the scripts for parsing CWL workflows, processing user commands, scheduling jobs using Airflow, and storing and accessing metadata in the metadata store (Datajoint [21]). All of this functionality is itself contained in a Docker container to simplify installation on the user’s machine. The CONDUIT container and related containers are started with Docker-compose.

The user interacts with CONDUIT through a series of command line tools. The user interface consists of:

- `conduit init`: used to configure AWS for cloud use through the provided cloudformation template. Optional for local use, and only needs to be run when configuring a new AWS account.
- `conduit build`: used to build the necessary tool Docker containers
- `conduit parse`: used to create a DAG from the CWL and schedule with Airflow. Accepts an optional parameterization file
- `conduit collect`: used to collect metadata results related to a workflow from the metadata database
- `conduit optimize`: used to schedule hyperparameter search for a given workflow

These commands provide the key method for users to schedule workflows, which can be monitored using the Apache Airflow webserver started with CONDUIT.

#### Workflow execution

The CONDUIT container shown in Fig. 6 provides SABER with a managed pipeline execution environment that can run locally or scale using the AWS Batch service. Our custom command scripts and CWL parser generate DAG specifications for execution by Apache Airflow. We select Apache Airflow to interface with a cloud-based computing solution. As an example, we use the AWS Batch service, although Airflow can interface with scalable cluster solutions such as Kubernetes or Hadoop. The framework facilitates the execution of a batch processing (versus streaming) workflow composed of software tools packaged inside multiple

software containers. This reduces the need to install and configure many, possibly conflicting software libraries.

### Cloud computation and storage

Large neuroimaging datasets are distinct from many canonical big data solutions because researchers typically analyze a few (often 1) very large datasets instead of many individual images. Custom storage solutions [20, 28, 29] exist but often require tools, knowledge, and access patterns that are disparate from those used by many neuroscience laboratories. SABER provides tools to connect to specialized neuroimaging databases that integrate into CWL tool pipelines. We use intern [57, 58] to provide access to bossDB and DVID and abstract data storage, RESTful calls, and access details. Workflow parameters, objective functions, and summary results such as graphs and cell densities can be stored using a DataJoint database [21] using a custom set of table schemas.

Some datasets, however, can be stored locally but are too large to process in memory on a single workstation. In addition to volumetric data stored in bossDB, SABER also supports local imaging file formats such as HDF5, PNG, or TIFF. As users share pipelines, they might wish to use a pipeline originally developed for data stored in one archive with that stored in another. Therefore, using the existing SABER tools raw and annotated data can be accessed, retrieved, and stored using:

- bossDB
- DVID
- Cloudvolume
- Amazon S3 buckets
- Local files (e.g., hdf5, numpy)

For intermediate results in a pipeline, files can be stored locally (or on any locally mounted drive) in numpy or HDF5 files or stored in AWS S3 buckets. Future work will increase the number of supported file formats. The modular nature of raw data access will allow additional tools to access new data sources as they emerge. Supporting further cloud systems will require additional development, although it will not affect the SABER tools or workflows. Currently only AWS is supported.

Modern cloud computing tools, such as AWS Batch or Kubernetes, allow large-scale deployment of containerized tools on demand. The CONDUIT container schedules workflows using Apache Airflow and currently supports 2 execution methods:

- AWS Batch
- Local compute resources

Workflows have a “local” flag, which can be set to indicate a choice of resources. Tools can also be configured to run with GPU resources. Both methods can be used with local or remote data storage. Further development will be required to enable support of further executors, such as Kubernetes, using the operators that exist in Apache Airflow.

### Required background and getting started

A new user to the SABER framework will require intermediate familiarity with Python programming, the use of command line tools (e.g., Bash), and Docker. These capabilities are often found in capable computer science undergraduates or new computationally oriented graduate students. To get started, new users will:

- Install Docker
- Build the desired tool containers (e.g., EM or X-ray containers) in the SABER folder
- Build and configure the core CONDUIT Docker containers
- Use the command line interface to schedule workflows

However, the use of SABER with the AWS cloud will require an AWS account, and  $\geq 1$  experienced AWS user to configure the system and serve as the administrator. To configure this system, the user needs to

- Use the cloudformation template to configure AWS Batch and S3
- Create credentials for other users and configure access from local machines

The envisioned users of this tool are neuroimaging laboratories, algorithm developers, and data analysts. One experienced user can quickly configure a cloud SABER deployment for use by others in the laboratory. Envisioned use cases include neuroimaging laboratories wanting to apply tools to newly collected datasets and tool developers who want to package and benchmark software tools to reach new users. While this framework certainly does not remove all barriers to entry, the use of Dockerized tools limits the number of competing software configurations for neuroimaging users and provides a common and powerful system for tool developers to share their work. Our system accomplishes this with a set of Dockerized tools to replace installing many, often conflicting dependencies with a single tool (i.e., Docker), the use of standard CWL definitions that are cross-compatible with other efforts, and specialized scripts to handle difficult use cases such as scheduling runs over large datasets using cloud computing resources. This approach attempts to balance the flexibility needed by tool developers with standardization to help the novice user. A user looking to deploy existing tools and workflows to new data will primarily interface through the user commands for CONDUIT, and a tool developer will primarily package tools following the Dockerfiles and CWL examples in SABER (to ensure compatibility with existing tools).

### Availability of Source Code and Requirements

The SABER framework is open source and available online:

- Project name: SABER
- Project home page: e.g., <https://github.com/aplbrain/saber>
- Operating system(s): Platform independent
- Programming language: Python, other
- Other requirements: Docker, AWS account (if scalable cloud computing required)
- License: Apache License 2.0
- RRID:SCR\_018812

### Data Availability

The source code for this project is available on GitHub, including code for tools and demonstration workflows. An extensive wiki documenting the repository is also hosted on GitHub. The data are stored in a bossDB instance at <https://api.bosssdb.io>. Snapshots of our code and other supporting data are openly available in the GigaScience repository, GigaDB [59].

## Abbreviations

AWS: Amazon Web Services; API: application programming interface; bossDB: Block and Object Storage Service Database; CWL: Common Workflow Language; DAG: directed acyclic graph; DVID: Distributed, Versioned, Image-Oriented Dataservice; EM: electron microscopy; fMRI: functional magnetic resonance imaging; GPU: graphics processing unit; GUI: graphical user interface; REST: representational state transfer; SABER: Scalable Analytics for Brain Exploration Research; XRM: X-ray microtomography.

## Competing Interests

The authors declare that they have no competing interests.

## Funding

Research reported in this publication was supported by the National Institute of Mental Health of the National Institutes of Health under Award No. R24MH114799 and Award No. R24MH114785. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health.

## Authors' Contributions

E.C.J.: conceptualization, investigation, formal analysis, methodology, software, supervision, and writing of the original draft. M.W.: investigation, software development, and methodology development. L.M.R.: investigation, software, data curation, visualization, review, and editing. R.N.T.: investigation, software, methodology, review, and editing. C.R.: conceptualization and software. N.D.: formal analysis and software development. D.K.: conceptualization, funding acquisition, and investigation. T.J.L.: software, resources, data curation, and validation. H.P.C.: software and visualization. J.D.: software and visualization. J.K.M.: conceptualization, software, and validation. M.J.H.: conceptualization, validation, investigation, and methodology. E.P.R.: conceptualization, validation, software, investigation, and methodology. B.A.W.: conceptualization, resources, funding acquisition, and project administration. E.L.D.: conceptualization, supervision, software, funding acquisition, project administration, investigation, review, and editing. K.P.K.: conceptualization, supervision, funding acquisition, project administration, review, and editing. W.R.G.R.: conceptualization, data curation, formal analysis, funding acquisition, investigation, methodology, project administration, software, supervision, and writing original draft.

## Acknowledgments

We thank the Apache Airflow and Common Workflow language teams for their open-source tools supporting reproducible workflows, as well as the research groups who produced our reference EM and XRM volumes for analysis.

## References

- Lichtman JW, Pfister H, Shavit N. The big data challenges of connectomics. *Nat Neurosci* 2014;17(11):1448.
- Micheva KD, O'Rourke N, Busse B, et al. Array tomography: High-resolution three-dimensional immunofluorescence. *Cold Spring Harb Protoc* 2010;5(11):1214–9.
- Chung K, Deisseroth K. CLARITY for mapping the nervous system. *Nat Methods* 2013;10(6):508–13.
- Allen Institute for Brain Science. Allen Brain Atlas. <http://brain-map.org/api/index.html>. Accessed on June 6, 2018.
- Dyer EL, Gray Roncal W, Prasad JA, et al. Quantifying mesoscale neuroanatomy using X-ray microtomography. *eNeuro* 2017;4(5), doi:10.1523/ENEURO.0195-17.2017.
- Plaza SM, Berg SE. Large-scale electron microscopy image segmentation in spark. *arXiv* 2016: 1604.00385.
- Knowles-Barley S, Kaynig V, Jones TR, et al. Rhoanet pipeline: Dense automatic neural annotation. *arXiv* 2016: 1611.06973.
- Gray Roncal WR, Kleissas DM, Vogelstein JT, et al. An automated images-to-graphs framework for high resolution connectomics. *Front Neuroinform* 2015;9:20.
- Berning M, Boergens KM, Helmstaedter M. SegEM: Efficient image analysis for high-resolution connectomics. *Neuron* 2015;87(6):1193–206.
- Matveev A, Meirovitch Y, Saribekyan H, et al. A multicore path to connectomics-on-demand. In: *Proceedings of the 22nd ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*. New York: ACM; 2017, doi:10.1145/3155284.3018766.
- Docker, Inc. Docker. <https://www.docker.com>. Accessed June 6, 2018.
- Common Workflow Language Working Group. Common Workflow Language. <https://www.commonwl.org>. Accessed on June 8, 2018.
- The Linux Foundation. Kubernetes. <https://kubernetes.io>. Accessed June 8, 2018.
- Pachyderm, Inc. Pachyderm. [www.pachyderm.io](http://www.pachyderm.io). Accessed June 8, 2018.
- Dask Development Team. Dask: Library for dynamic task scheduling. 2018 <https://dask.org>. Accessed on September 10, 2018.
- Kiar G, Brown ST, Glatard T, et al. A serverless tool for platform agnostic computational experiment management. *arXiv* 2018, 1809.07693.
- Apache. Airflow. <https://airflow.apache.org>. Accessed June 8, 2018.
- UCSC Computational Genomics Lab. TOIL. 2018. <http://toil.ucsc-cgl.org>. Accessed June 8, 2018.
- Kotliar M, Kartashov A, Barski A. CWL-Airflow: A lightweight pipeline manager supporting Common Workflow Language. *Gigascience* 2019;8(7), doi:10.1093/gigascience/giz084.
- Hider R, Kleissas DM, Pryor D, et al. The Block Object Storage Service (bossDB): A cloud-native approach for petascale neuroscience discovery. *bioRxiv* 2017, doi:10.1101/217745.
- Vathes LLC. Datajoint: A hub for developing, sharing, and publishing scientific data pipelines. <https://datajoint.io>. Accessed September 10, 2018.
- Bock DD, Lee WCA, Kerlin AM, et al. Network anatomy and in vivo physiology of visual cortical neurons. *Nature* 2011;471(7337):177.
- Kasthuri N, Hayworth K, Berger D, et al. Saturated reconstruction of a volume of neocortex. *Cell* 2015;162(3): 648–61.
- Takemura Sy, Bharioke A, Lu Z, et al. A visual motion detection circuit suggested by *Drosophila* connectomics. *Nature* 2013;500(7461):175.
- Lee WCA, Bonin V, Reed M, et al. Anatomy and function of an excitatory network in the visual cortex. *Nature* 2016;532(7599):370.

26. Hieber SE, Bikis C, Khimchenko A, et al. Tomographic brain imaging with nucleolar detail and automatic cell counting. *Sci Rep* 2016;6, doi:10.1038/srep32156.
27. Busse M, Müller M, Kimm MA, et al. Three-dimensional virtual histology enabled through cytoplasm-specific X-ray stain for microscopic and nanoscopic computed tomography. *Proc Natl Acad Sci U S A* 2018;115(10):2293–98.
28. Plaza S, Katz W. DVID. <https://github.com/janelia-flyem/dvid>. Accessed June 6, 2018.
29. Seung Lab. cloud-volume. 2018. <https://github.com/seung-lab/cloud-volume>. Accessed June 6, 2018.
30. Gürsoy D, De Carlo F, Xiao X, et al. TomoPy: A framework for the analysis of synchrotron tomographic data. *J Synchrotron Radiat* 2014;21(5):1188–93.
31. Sommer C, Straehle C, Koethe U, et al. Ilastik: Interactive learning and segmentation toolkit. In: *Biomedical Imaging: From Nano to Macro*. IEEE; 2011:230–33.
32. Ronneberger O, Fischer P, Brox T. U-net: Convolutional Networks for Biomedical Image Segmentation. Springer; 2015.
33. Ciresan D, Giusti A, Gambardella LM, et al. Deep neural networks segment neuronal membranes in electron microscopy images. In: *Advances in Neural Information Processing Systems*. 2012:2843–51.
34. Gray Roncal W, Pekala M, Kaynig-Fittkau V, et al. VESICLE: Volumetric evaluation of synaptic interfaces using computer vision at large scale. *arXiv* 2014: 1403.3724.
35. Staffler B, Berning M, Boergens KM, et al. SynEM, automated synapse detection for connectomics. *eLife* 2017;6, doi:10.7554/eLife.26414.
36. Nunez-Iglesias J, Kennedy R, Plaza SM, et al. Graph-based active learning of agglomeration (GALA): A Python library to segment 2D and 3D neuroimages. *Front Neuroinform* 2014;8:34.
37. Nunez-Iglesias J, Kennedy R, Parag T, et al. Machine learning of hierarchical clustering to segment 2D and 3D images. *PLoS One* 2013;8(8):e71715.
38. Johnson EC, Rodriguez LM, Norman-Tenazas R, et al. Transfer learning analysis of image processing workflows for electron microscopy datasets. In: *2019 53rd Asilomar Conference on Signals, Systems, and Computers*. IEEE; 2019:1197–201.
39. Bergstra J, Bengio Y. Random search for hyper-parameter optimization. *J Mach Learn Res* 2012;13:281–305.
40. Jones DR, Schonlau M, Welch WJ. Efficient global optimization of expensive black-box functions. *J Glob Optim* 1998;13(4):455–92.
41. Azar MG, Dyer E, Kording K. Convex relaxation regression: Black-box optimization of smooth functions by learning their convex envelopes. *arXiv* 2016, 1602.02191.
42. bossDB Projects. <https://bosddb.org/projects>. Accessed May 10, 2020.
43. Prasad JA, Balwani AH, Johnson EC, et al. A three-dimensional thalamocortical dataset for characterizing brain heterogeneity. *Sci Data* 2020;7, doi:10.1038/s41597-020-00692-y.
44. Google. Neuroglancer. 2018. <https://github.com/google/neuroglancer>. Accessed June 6, 2018.
45. Gorgolewski K, Burns CD, Madison C, et al. Nipype: A flexible, lightweight and extensible neuroimaging data processing framework in python. *Front Neuroinform* 2011; 5:13.
46. Matelsky J. Substrate. <https://github.com/iscoe/substrate>. Accessed June, 8 2018.
47. LaGrow TJ, Moore MG, Prasad JA, et al. Approximating cellular densities from high-resolution neuroanatomical imaging data. *Annu Int Conf IEEE Eng Med Biol Soc* 2018, doi:10.1109/EMBC.2018.8512220.
48. Kiar G, Gorgolewski KJ, Kleissas D, et al. Science In the Cloud (SIC): A use case in MRI connectomics. *Gigascience* 2017;6(5), doi:10.1093/gigascience/gix013.
49. Apache. Apache Hadoop. <https://hadoop.apache.org>. Accessed June, 8 2018.
50. Dinov I, Van Horn J, Lozev K, et al. Efficient, distributed and interactive neuroimaging data analysis using the LONI pipeline. *Front Neuroinform* 2009;3:22.
51. Afgan E, Baker D, Van den Beek M, et al. The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2016 update. *Nucleic Acids Res* 2016;44(W1):W3–W10.
52. Berthold MR, Cebron N, Dill F, et al. KNIME-the Konstanz information miner: version 2.0 and beyond. *ACM SIGKDD Explor Newsl* 2009;11(1):26–31.
53. Glatard T, Ferreira da Silva R, Nouha B, et al. Boutiques: An application-sharing system based on Linux containers. *Front Neurosci* 2015, 9, doi:10.3389/conf.fnins.2015.91.00012.
54. CenturyLink Labs. DRAY: Docker Workflow Engine. <https://github.com/CenturyLinkLabs/dray>. Accessed June 8, 2018.
55. Air-tasks. <https://github.com/wongwill86/air-tasks>. Accessed June 8, 2018.
56. SABER. <https://github.com/aplbrain/saber>. Accessed May 10, 2020.
57. Matelsky J. Intern: Integrated Toolkit for Extensible and Reproducible Neuroscience. <https://github.com/jhuapl-boss/intern>. Accessed June 10, 2018.
58. Matelsky JK, Rodriguez L, Xenos D, et al. Intern: Integrated toolkit for extensible and reproducible neuroscience. *bioRxiv* 2020, doi:10.1101/2020.05.15.098707.
59. Johnson EC, Wilt M, Rodriguez L, et al. Supporting data for “A scalable framework for reproducible processing of volumetric, nanoscale neuroimaging datasets.” *GigaScience Database* 2020. <http://dx.doi.org/10.5524/100822>.