GigaScience

Toward A Scalable Framework for Reproducible Processing of Volumetric, Nanoscale Neuroimaging Datasets

--Manuscript Draft--

Manuscript Number:	GIGA-D-19-00133R1
Full Title:	Toward A Scalable Framework for Reproducible Processing of Volumetric, Nanoscale Neuroimaging Datasets
Article Type:	Technical Note
Funding Information:	National Institutes of Health (R24MH11479) — Dr. William Gray-Roncal
Abstract:	Background: Emerging neuroimaging datasets (collected with imaging techniques such as Electron Microscopy, Two-Photon Calcium Imaging, 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 utilize 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, that 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.
Corresponding Author:	Erik C. Johnson, Ph.D. Johns Hopkins University Applied Physics Laboratory UNITED STATES
Corresponding Author Secondary Information:	
Corresponding Author's Institution:	Johns Hopkins University Applied Physics Laboratory
Corresponding Author's Secondary Institution:	
First Author:	Erik C. Johnson, Ph.D.
First Author Secondary Information:	
Order of Authors:	Erik C. Johnson, Ph.D. Miller Wilt Luis M. Rodriguez Raphael Norman-Tenazas Corban Rivera Nathan Drenkow Dean Kleissas Theodore J. LaGrow Hannah P. Cowley

	Joseph Downs
	Jordan Matelsky
	Marisa Hughes
	Elizabeth Reilly
	Brock Wester
	Eva Dyer
	Konrad Kording
	William Gray-Roncal
Order of Authors Secondary Information:	
Response to Reviewers:	Dear Gigascience Editors, We would like to thank the reviewers and editor for their thoughtful and thorough review of our manuscript, and we are happy to present our revisions to the manuscript. Below, we attempt to highlight how these revisions respond to specific concerns from the reviewers, and improve the quality of the submission accordingly. We have also submitted our tool “Scalable Analytics for Brain Exploration Research” to scicrunch.org, with the resulting RRID:SCR_018812, which has been added to the manuscript. We look forward to hearing your feedback on this improved manuscript. We apologize for the delay in providing this response and appreciate your understanding. Thank you very much, Erik C. Johnson, on behalf of all authors Detailed Response: Reviewer 1 raised some concerns about the manuscript. We highlight these, along with our response and changes. In addition to the responses below, we made several minor modifications, made capitalization consistent, and changed the color scheme of the table in response to Reviewer 1’s comments. Comment: “I think the reproducibility aspect relates to the processing, not to the reproducibility of the framework. The generality of the term "neuroimaging" had me anticipate a truly generic solution. However, the manuscript itself is focused on a specific subset of data modalities, with specific flavors of being "large". It is unclear to me, if the proposed solution is capable and/or meaningful to be applied to other neuroimaging data, such as MRI, PET, MEG, etc. It would be good to clarify that directly in the title/abstract.” Response: We agree with the reviewer and do not wish to mislead the readers as to the applicability of our software framework. We propose modifying the title to “Toward A Scalable Framework for Reproducible Processing of Volumetric, Microscale and Nanoscale Neuroimaging Datasets” To clarify the neuroimaging modalities that work well in this framework (EM, X-ray Microtomography, and Light microscopy) that are best suited for this framework. This is also discussed in a new paragraph in the discussion section. We additionally note in the discussion that this framework is suitable for large-scale batch processing in many domains, but that our specific use-cases focus as described above. Require technical Expertise and deployment Comment: “The paper poses the lack of computational expertise and resources as a major challenge regarding large-scale data analysis. I personally support this claim. However, it is not clear to me, if the proposed solution can successfully solve this problem in its entirety, or to which degree. In some sense, the proposed solution is assembled from a substantial number of extremely capable, but also rather complex technologies. Therein lies both potential and difficulty regarding the "lack of expertise". The manuscript, in its present form, does not convince me that SABER/CONDUIT can indeed _lower_ the technical threshold, as opposed to presenting a different set of

technical challenges when compared to other solutions. It would be instrumental to know what exactly the path to adoption for a research group (would) look like."

Response: The reviewer has made a great point. We have not conducted extensive usability studies to report here. Anecdotally, however, we believe this is some merit to this claim, particularly in the Electron Microscopy community. This community has repeatedly struggled due to tools which are difficult to install, use and compare. Users need to learn unrelated and often incompatible software libraries and interfaces. While the SABER framework does not address all issues reducing adoption of new tools, we do believe we can contribute in addressing standardization and compatibility issues. This space is severely underpopulated in terms of interoperable technologies and solutions.

Specifically, we believe the standardized library described in the section "Standardized Workflows and Tools" does help address the required background skills and experience required to run different tools required by the community. It provides an opportunity and a mechanism for labs and tools to work together.

However, the reviewer's point is still well taken. We have added a section "Required Background and Getting Started" that addresses the required path for adoption and computational expertise still required to utilize this framework. We hope that future work (now discussed in the section "Potential implications") will integrate this framework into existing datastores and provide graphical user interface tools to enable users with limited.

Comment: "Is it AWS only? must data be in BossDB? What if these 3rd-party services are discontinued?"

Response: The computation framework is flexible in terms of backend computation, due to the use of the Apache Airflow project, although currently local docker execution and AWS batch are supported. We hope additional resource schedulers will be added as the community requires/demands, and the software is released open source for others to adapt as they wish.

For storage, files can be stored locally (or on any locally mounted drive) in numpy or hdf5 files, stored on AWS S3, or stored in the BossDB or DVID systems. This is clarified in the "Cloud Computation and Storage" section in "Methods" in response to this review.

The dependence on 3rd party solutions (AWS, Airflow, bossDB, Datajoint) is certainly a risk if these technologies fail to be supported and developed successfully. However, we believe the strength of building on these existing tools outweighs this risk. For storage and computation, this risk is somewhat mitigated by a modular approach that allows for different operators and datastores. A short discussion of this limitation has been added to the "Discussion" section.

Comment: "what technical expertise needs to exist in a group that aims to adopt this solution?"

Response: Generally speaking, the group needs experience in python and the use of docker containers. Experience with linux systems is preferable, and an AWS account is currently required for scalable computing (although not required if using local resources). While these are non-trivial, we believe that this experience is becoming more widespread (e.g., CS undergraduate student) and we are continuing to lower the barrier to entry through GUI development and improved training tools.
In response to this review, a new section "Required Background and Getting Started" has been added to address this issue as it currently stands in this manuscript.

Comment: "what deployments already exist? Or in other words: what empirical evidence exists that intellectual merits of the proposal translate into practical advantages?"

Response: The SABER platform is an emerging research tool, and this manuscript will help us to socialize the tools with the broader community. We do actively develop and

share workflows using this platform for many of our heterogenous research products and especially with our collaborators at Georgia Tech. We have enabled generation of new data products such as synapse labels, segmentations, and cell density analysis as well as benchmarking of algorithms in the projects listed below. We are working on widespread community adoption, but have not reached that point yet, which is a fair criticism - and hopefully enabled by this manuscript.

We are integrating our solution with the bossDB access portal (http://bossdb.org/) to process the datasets hosted there to help create an ecosystem for the community. Our collaborators at Georgia Tech are using these techniques for processing X-ray microtomography data. The underlying tools that we leverage in our platform are used in many applications.

Example uses of the tools so far include:
- Unsupervised learning pipelines for EM processing (https://ieeexplore.ieee.org/abstract/document/9048673)
- Processing pipelines for new xray microtomography datasets (https://www.biorxiv.org/content/10.1101/2020.05.22.111617v1.abstract)
- Benchmarking novel optimization approaches for workflows (https://arxiv.org/abs/2006.02624)
- NIH Brain Initiative Symposium, poster, and demonstrations (Symposium 5) https://www.labroots.com/ms/virtual-event/2020-6th-annual-brain-initiative-investigators-virtual-meeting/agenda

Details, open source development and interfacing with other systems:

Comment: The manuscript contrasts SABER/CONDUIT with other tools like LONI and Nipype, and advertises that it frees researchers from being locked into specific idiosyncrasies of these tools and their limitations (need a cluster/no cloud, etc.), and describes an "ecosystem of neuroimaging data analysis pipelines". However, it is unclear from the manuscript in how SABER/CONDUIT does this better than the rest.

Response: While existing pipeline tools like LONI and Nipype enable the execution of scientific workflows, they are lacking a few key features for the neuroimaging user.

First is the library of tools required for modern segmentation and detection problems on EM data, including GPU enabled DNN tools. Second is the orchestration of docker containers, which enable tools with conflicting software dependencies and operating systems. Third are tools to deploy this easily over a single large dataset (parameterize jobs over a single large dataset, optimize multistage workflows end to end). We believe these key features will encourage adoption compared to existing tools.

We have attempted to address this comment by adding additional clarification to the caption of Table 1, as well as to the final two paragraphs of “Existing Software Solutions” in the “Methods” section.

Comment: “ how open is that ecosystem? What is needed to contribute to it? The repository at https://github.com/aplbrain/saber only shows two dozen commits, most done very recently, and not a single pull request. Is this the place where core development is taking place?”

Response: The reviewer raises a great question given the relative newness of the project. The project was released open source just prior to submission of the manuscript. Contributors are now openly developing, creating pull requests, and issues addressed by the team. While the list of contributors is small, we are growing the team. This is now the current repo for core development. We hope momentum will continue to build over time as more users and contributors join the project.

Comment: “which aspects/components of the proposed solution are generic vs specific to the demo'ed data analyses and modalities?”

Response: The reviewer is right that to ensure broad impact, the system needs to be as general as possible. The tools and pipelines are developed for volumetric datasets like EM and XRM; although they perform repeated (e.g., batch) generic problems like

object detection, classification, and 2D and 3D segmentation, the parameters and weights provided are specific to these modalities. Users can adapt them to their own problem using the provided optimization framework, but may require annotated data.

The backend code for execution of CWL pipelines of docker containers over large datasets is generic and applicable to any processing pipelines applied over volumetric data.
We have attempted to address this concern through additions in the section “Pipelines and Tools for Neuroimaging Data” as well as with a new paragraph in the “Discussion” section.

Comment: why is CWL better than, say, a workflow definition in Python, as done by Nipype? I do believe that it is, and other projects are moving in this direction, but the manuscript does not make much of a point in this regard.

Response: The reviewer raises a great point about the use of CWL compared to other approaches. We believe that the common, interoperable standard is important to 1) allow reuse of the pipelines in other systems 2) allow pipelines developed for other open source project to be deployed using the SABER/CONDUIT system and 3) reduces the need to yet another workflow specification specific to the system, as is typically done in python workflow managers. This approach leverages the work done by the CWL community.
We have added a paragraph discussing the strengths of the CWL approach to the “Discussion” section.

System Design:
Comment: The manuscript left me with a number of open questions regarding the design specifics of the proposed system. I would personally much prefer to have the manuscript parts concerning example analyses (Fig 1, 2, 3, 4, 5) condensed in favor of a more concrete description of the frameworks components (methods, and Fig. 6). How do they interact (APIs, etc)? What motivated the specific choice of these components? Can they be replaced with alternatives, for example, to better fit into existing institutional infrastructure? The ability to interface "abstract data storage" is briefly mentioned, but there are no specifics regarding required properties of such a storage system.

Response: Thank you for this point. To address this comment we have expanded the discussion of the framework and components, and the discussion of Figure 6.
We have tried to address:
1. API definition
2. Design choices for each component
3. Alternative approaches for different components

We have also addressed the comment on abstract data storage. We hope these changes address the reviewer’s concerns. To address these concerns, we have made extensive revisions to the section “Framework Components” on the components, design choices, and APIs.

Comment: Fig. 6 seems to suggest that there is much flexibility in how this system can be made to work, but it remains unclear to me, which components are, e.g., cloud-only. Moreover, I am unsure about the relationship of SABER and CONDUIT (which both seems to live in the same code repository). Fig. 6 depicts it, congruent with the manuscript, as a central system components. However, it is only very briefly described in the methods section. It seems to me that SABER users would actually mostly interact with CONDUIT, while SABER comprises (in addition) the execution and data access infrastructure. What exactly is CONDUIT adding to cwl-airflow?

Response: Thank you for pointing out these confusing issues. The CONDUIT core is the python code and scripts that build upon CWL and airflow. This includes parsing the CWL and deploying DAGs, as in cwl-airflow. Features added on top of this basic functionality include parameterizing jobs for deployment over chunks of data (specified by coordinates), iterative execution of the same DAG with different parameters (for parameter optimization), and logging of metadata and job results. Moreover, wrappers for the use of local files and s3 files for intermediate results with the same workflows. These features are critical for our machine learning pipeline deployment.

SABER is the collection of tools and workflows, including docker containers and python code for data access, processing, and so forth. A user looking to deploy tools to new data would primarily work with CONDUIT, and a tool developer primarily work with SABER (to ensure compatibility with existing tools). These changes can be seen in Figure 6, as well as in the subsection SABER of the section Methods.

Reviewer 2
Reviewer 2 also raised several key points about the manuscript, which we have responded to; these resulted in significant improvements to the manuscript.

Comment: The authors should identify the likely users of SABER/CONDUIT. For example, would these be labs new to neuroimaging analysis or would they be users with experience who are working with already functioning software pipelines and want more modularity? What are the likely barriers to adoption for the community of interest, and how has this software solution been created so as to minimize these barriers?

Response: This is a great point, thanks for this insight. We imagine this would be established neuroimaging labs with some analysis experience. Our envisioned target users are:
1. Neuroimaging labs wanting to apply tools to new collected datasets to segment data, detect objects of interest, and perform analysis
2. Tool developers who want to package tools to reach new users
We have found that neuroimaging labs have a lot of issues with getting new software tools working on their systems, and deploying those tools at scale. Often, this never happens.
Our solution addresses this issue through several features.
1.A set of dockerized tools to replace installing many, often conflicting dependencies with a single tool (docker)
2.Use of standard CWL definitions which are cross-compatible with other efforts
3.Specialized scripts to handle scheduling runs over large datasets using cloud computing resources
The solution attempts to balance the flexibility needed by tool developers with standardization to help the novice user.
We have added a section “Required Background and Getting Started” to address these issues.

Comment: Cell identification and connectome maps are generated independently from two distinct tools and datasets (X-ray and EM, respectively). Presumably, many users would be interested in integrating the two outcomes, however no explicit connection is drawn or discussed regarding how complementary outcomes can be correlated and analyzed together through this software.

Response: The reviewer raises a good point- this is in fact possible in this framework. The tools and datasets can be combined in several ways using CWL and docker containers within our framework.
1.The same tools can be used for different steps of both workflows. For instance, our U-net tool can be used to generate probability maps for synapses, cell bodies, or cell membranes when training with different data.
2.Co-registered datasets can be jointly analyzed using our CWL pipelines, and results can all be stored in the bossDB system. This allows the user to use simple python scripts to pull and analyze any parts of these data.
We have added language discussing this to the second paragraph of the discussion.

Comment: In general, it would be useful for the software to include example data that are fully analyzed, together with a mechanism for comparison with user datasets. This would help to identify how datasets from different sources vary, for example for determining the impact of data collection methodology and format, as opposed to differences in the biological system under study.

Response: This is a great point. We have provided some previously processed and existing datasets. First, the fully analyzed datasets in the figures are included with gigascience submission for existing figures.
Moreover, many raw data and processed output datasets are hosted at https://bossdb.org/projects/ and accessible using our bossDB access tools. Users can

	quickly modify workflows to run on their data and pull this reference data from bossDB for comparison. This should allow users to make these comparisons, combined with our existing metrics code from our workflows. We have added a section “Datasets for Benchmarking Workflows” to address this. Comment: The section entitled "Optimization and Deployment of Workflows" is vague and generally unclear. Response: Thank you for the feedback, we’ve attempted to expand the details in this section to better explain the purpose and the solution. We have revised this section to clearly list the features which enable 1) large-scale deployment of the workflows in the proceeding section and 2) fine-tuning of workflow hyperparameters for deployment to new datasets. We have explicitly listed these use cases and the features which enable them, and we hope this clarifies the need for this section. Comment: The "Existing Software Solutions" section seems more appropriate for the Background section. Response: Thank you for the organizational suggestion, while this is certainly important background, we preferred to avoid this detailed discussion of specific software features in the introduction, which we saved for the detailed discussions of software architecture and features later in the manuscript. The point, however, is taken and we have expanded the paragraph in the Introduction beginning with “In other domains, computer science solutions exist for improving algorithm portability and reproducibility” to capture the key features of this information and reference the table. Comment: The Table 1 caption includes extensive discussion that should be in the main text of the document, rather than the caption. There is little discussion of this table in general, although it is quite valuable. Response: Thank you for this suggestion, we are glad the table provides a useful comparison of features. We expanded the discussion of this table in “Existing Software Solutions” to provide key context in comparing the features of SABER with other pipeline approaches. Comment: “In Figure 5, it would be good to include the raw image and to color-code the membranes and synapses for clarity, rather than using black and white.” Response: Thank you for this suggestion for color coding. We used the standard tool neuroglancer (https://github.com/google/neuroglancer) to generate this plot, and wanted to remain consistent with the visualization hosted online, e.g. at bossdb.org. Probability maps in this format are not visualized with colors (synapses and membranes), whereas annotation ids are. We hope this explains the formatting of the images to be consistent with the existing tools, e.g. the visualizations of data hosted at https://bossdb.org/projects/. We have noted this in the figure caption.
Additional Information:	
Question	**Response**
Are you submitting this manuscript to a special series or article collection?	No
Experimental design and statistics Full details of the experimental design and statistical methods used should be given in the Methods section, as detailed in our Minimum Standards Reporting Checklist. Information essential to interpreting the data presented should be made available	Yes

in the figure legends. Have you included all the information requested in your manuscript?	
Resources A description of all resources used, including antibodies, cell lines, animals and software tools, with enough information to allow them to be uniquely identified, should be included in the Methods section. Authors are strongly encouraged to cite Research Resource Identifiers (RRIDs) for antibodies, model organisms and tools, where possible. Have you included the information requested as detailed in our Minimum Standards Reporting Checklist?	Yes
Availability of data and materials All datasets and code on which the conclusions of the paper rely must be either included in your submission or deposited in publicly available repositories (where available and ethically appropriate), referencing such data using a unique identifier in the references and in the “Availability of Data and Materials” section of your manuscript. Have you have met the above requirement as detailed in our Minimum Standards Reporting Checklist?	Yes

```
This is pdfTeX, Version 3.14159265-2.6-1.40.19 (TeX Live 2018/W32TeX)
(preloaded format=pdflatex 2018.7.12)  19 AUG 2020 17:05
entering extended mode
 restricted \write18 enabled.
 %&-line parsing enabled.
**final_revisions.tex
(./final_revisions.tex
LaTeX2e <2018-04-01> patch level 5

! LaTeX Error: File `oup-contemporary.cls' not found.

Type X to quit or <RETURN> to proceed,
or enter new name. (Default extension: cls)

Enter file name:
! Emergency stop.
<read *>

l.11 ^^M

*** (cannot \read from terminal in nonstop modes)


Here is how much of TeX's memory you used:
 10 strings out of 492646
 259 string characters out of 6133325
 56709 words of memory out of 5000000
 3994 multiletter control sequences out of 15000+600000
 3640 words of font info for 14 fonts, out of 8000000 for 9000
 1141 hyphenation exceptions out of 8191
 10i,0n,8p,97b,8s stack positions out of 5000i,500n,10000p,200000b,80000s
!  ==> Fatal error occurred, no output PDF file produced!
```

OXFORD

(GIGA)n SCIENCE

GigaScience, 2017, 1–13

doi: xx.xxxx/xxxx
Manuscript in Preparation
Tech Note

TECH NOTE

Toward A Scalable Framework for Reproducible Processing of Volumetric, Nanoscale Neuroimaging Datasets

Erik C. Johnson[1,*], Miller Wilt[1], Luis M. Rodriguez[1], Raphael Norman-Tenazas[1], Corban Rivera[1], Nathan Drenkow[1], Dean Kleissas[1], Theodore J. LaGrow[2], Hannah Cowley[1], Joseph Downs[1], Jordan Matelsky[1], Marisa Hughes[1], Elizabeth Reilly[1], Brock Wester[1], Eva Dyer[2], Konrad Kording[3] and William Gray-Roncal[1,*]

[1]Johns Hopkins University Applied Physics Laboratory; Laurel, Maryland and [2]Georgia Institute of Technology; Atlanta, Georgia and [3]University of Pennsylvania; Philadelphia, Pennsylvania

[*]erik.c.johnson@jhuapl.edu; william.gray.roncal@jhuapl.edu

Abstract

Background Emerging neuroimaging datasets (collected with imaging techniques such as Electron Microscopy, Two-Photon Calcium Imaging, 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 utilize 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, that 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.

Key words: Computational Neuroscience; Microtomography; Electron Microscopy; Workflows; Containers; Optimization; Reproducible science

Introduction

Testing modern neuroscience hypotheses often requires robustly processing large datasets. Often the labs 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

Compiled on: August 19, 2020.
Draft manuscript prepared by the author.

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, 7, 8, 9, 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 three 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 offers unique insight into nanoscale connectivity [22, 23, 24, 25], and XRM allows for rapid assessment of cells and blood vessels at scale [26, 5, 27]. These approaches provide complementary information and have been successfully used on the same biological sample [5], as 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 being able 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 impacted 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 1) Dockerized tools exist, 2) CWL workflows can be specified, and 3) 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 like 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 neuroimagery datasets, these tools are frequently lab 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 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 users need to be familiar with, helping to limit the computational experience required to run these pipelines.

Initially, we have implemented two 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 Microcscopy
XRM provides a rapid approach for producing large-scale submicron 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. Cubic millimeter-sized samples (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 implementation 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. Figure 1 shows this canonical workflow for XRM data, with each block representing a separate containerized tool. Also shown in Panel B 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 retraining) 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.

Deriving Synapse-level Connectomes from Electron Microscopy

Several workflows exist to produce graphs of brain connectivity from EM data [6, 10, 7], 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 Figure 2. First, data is 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 more than one step of the pipeline. Examples of tools for membrane segmentation include CNN [33] and U-nets [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, 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 algorithms developers. For scientists, this provides an established and tested pipeline for initial discovery. For algorithm developers, this pipeline can be used to benchmark algorithms which encompass one or more steps 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 1) a paramterization API to distribute jobs over large volumes of data and 2) an optimization API to train pipelines and fine-tune hyperparameters for new datasets.

To apply SABER workflows to large volumetric datasaets, 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.

In order to tune SABER workflows for new datasets, it in necessary to train the parameters of the pipeline, including any hyperparameter optimization (Figure 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 enables iteratively selecting parameters, scheduling parallel jobs, and collecting 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 Figure 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 syatem ([20],https://bossdb.org/projects) for this purpose. For testing XRM pipelines, data from the datasets “Dyer et al. 2016” [5] and “Prasad et al. 2020” [42] can be used. These datasets contain different brain regions including labels of cell bodies and vasculature for training new users, 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 collecting a new neuroimaging dataset, 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 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. Figure 3 demonstrates the application of three algorithms for pipeline optimization. We choose the Allen Institute for Brain Science (AIBS) Reference Atlas [4] as a demonstration of generalization beyond EM and XRM datasets. In order to optimize the pipeline, this example optimizes over: 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 Figure 3. Random selection of parameters often produces comparable results to 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. Figure 3B shows a parameter reduction of twenty percent 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 Figure 4, and another example of this use case for extracting synapse-level connectomes can be seen in Figure 5.

Use Case 3: Benchmarking Neuroimaging Algorithms

The third major use case applies to developers implementing new algorithms for neuroimaging datasets. Due 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 which 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 one 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, PET, 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 which 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 which 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, 43]. 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 open source packages, substrate [44] and neuroglancer [45], to easily visualize data inputs and outputs of our workflows and tools.

Scalable solutions for container 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 usecases. 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 3rd 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 like light microscopy [46], 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, demos, 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 available to the public with the expectation 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], [45]). In an effort to lower the barriers for new users, this work will include Graphical User Interfaces (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 which are fully portable and reproducible have been produced (e.g., [47]), 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 [48], 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 backend for workflow management systems such as SABER.

Traditional workflow environments (e.g., LONI Pipeline [49], Nipype [43], Galaxy [50], and Knime [51]) 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 are 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 [52] 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 [53] 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 light-weight 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 [54], 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 which are not provided in these more general purpose scientific workflow approaches due to the use of volumetric data, few large datasets (vs. 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 [50], 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 [49] and Nipype [43] enable the execution of scientific workflows, they are still lacking a few key features for the neuroimaging user and may limit the portability and utility of workflows. SABER provides a the library of tools required for modern segmentation and detection problems on EM and XRM data, including GPU enabled DNN tools. These tools, and their corresponding CWL definitions, are useful in any system which 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.

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 neuroimag-

ing 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 two 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 (Figure 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 graphical user interface (GUI). Each tool used in the workflows will also be built into a separate image.

In our CONDUIT framework (Figure 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 reconfiguration.

The repository at `github.com/aplbrain/saber` contains both our CONDUIT framework and the SABER workflows and tools, as is visualized in Figure 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.

SABER Workflow Library

The SABER subproject consists of a library of code, tools, and workflows. Each SABER tool much 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 Directed Acyclic Graph 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 hyperparamter 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 Figure 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 utilize 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 conflict-

ing software libraries.

Cloud Computation and Storage

Large neuroimaging datasets are distinct from many canonical big data solutions because researchers typically analyze a few (often one) 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 which integrate into CWL tool pipelines. We use intern [55, 56] 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 (hdf5, numpy, etc)

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 two 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 which 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 at least one 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 labs, algorithm developers, and data analysts. One experienced user can quickly configure a cloud SABER deployment for use by others in the lab. Envisioned use cases include neuroimaging labs 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 which 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

Availability of supporting data and materials

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.bossdb.org.

Declarations

List of abbreviations

EM- Electron Microscopy, XRM- X-Ray Microtomography, AWS- Amazon Web Services, CWL- Common Workflow Language DVID- Distributed, Versioned, Image-Oriented Dataservice, bossDB- Block and Object Storage Service Database, SABER- Scalable Analytics for Brain Exploration Research, DAG- Directed Acyclic Graph, CNN- Convolutional Neural Net-

work, AIBS- Allen Institute for Brain Science

Competing Interests

The author(s) 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 Number R24MH114799. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health.

Author's 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.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.M.: conceptualization, software, and validation. M.H.: Conceptualization, validation, investigation, and methodology. E.R.: conceptualization, validation, software, investigation, and methodology. B.W.: conceptualization, resources, funding acquisition, and project administration. E.D.: conceptualization, supervision, software, funding acquisition, project administration, investigation, review and editing. K.K.: conceptualization, supervision, funding acquisition, project administration, review and editing. W.G.R.: conceptualization, data curation, formal analysis, funding acquisition, investigation, methodology, project administration, software, supervision, writing original draft.

Acknowledgements

We would like to 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

1. Lichtman JW, Pfister H, Shavit N. The big data challenges of connectomics. Nature neuroscience 2014;17(11):1448.
2. Micheva KD, O'Rourke N, Busse B, Smith SJ. Array tomography: High-resolution three-dimensional immunofluorescence. Cold Spring Harbor Protocols 2010;5(11):1214–1219.
3. Chung K, Deisseroth K. CLARITY for mapping the nervous system. Nature methods 2013 jun;10(6):508–13.
4. Allen Institute for Brain Science, Allen Brain Atlas; Retrieved June 2018. Http://brain-map.org/api/index.html.
5. Dyer EL, Roncal WG, Prasad JA, Fernandes HL, Gürsoy D, De Andrade V, et al. Quantifying mesoscale neuroanatomy using X-ray microtomography. eNeuro 2017;4(5):ENEURO–0195.
6. Plaza SM, Berg SE. Large-scale electron microscopy image segmentation in Spark. arXiv preprint 2016;.
7. Knowles-Barley S, Kaynig V, Jones TR, Wilson A, Morgan J, Lee D, et al. RhoanaNet pipeline: Dense automatic neural annotation. arXiv preprint 2016;.
8. Gray Roncal WR, Kleissas DM, Vogelstein JT, Manavalan P, Lillaney K, Pekala M, et al. An automated images-to-graphs framework for high resolution connectomics. Frontiers in neuroinformatics 2015;9:20.
9. Berning M, Boergens KM, Helmstaedter M. SegEM: efficient image analysis for high-resolution connectomics. Neuron 2015;87(6):1193–1206.
10. Matveev A, Meirovitch Y, Saribekyan H, Jakubiuk W, Kaler T, Odor G, et al. A multicore path to connectomics-on-demand. In: Proceedings of the 22nd ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming ACM; 2017. p. 267–281.
11. Docker, Inc , Docker; Retrieved June 2018. Https://www.docker.com.
12. working group CWL, Common Workflow Language; Retrieved June 2018. Https://www.commonwl.org.
13. Foundation TL, Kubernetes; Retrieved June 2018. Https://kubernetes.io.
14. Pachyderm I, Pachyderm; Retrieved June 2018. Www.pachyderm.io.
15. Dask Development Team. Dask: Library for dynamic task scheduling; 2016, https://dask.org.
16. Kiar G, Brown ST, Glatard T, Evans AC. A Serverless Tool for Platform Agnostic Computational Experiment Management. CoRR 2018;abs/1809.07693. http://arxiv.org/abs/1809.07693.
17. Apache, Airflow; Retrieved June 2018. Https://airflow.apache.org.
18. Lab UCG, TOIL; Retrieved June 2018. Http://toil.ucsc-cgl.org.
19. Kotliar M, Kartashov A, Barski A. CWL-Airflow: a lightweight pipeline manager supporting Common Workflow Language. bioRxiv 2018;p. 249243.
20. Kleissas D, Hider R, Pryor D, Gion T, Manavalan P, Matelsky J, et al. The Block Object Storage Service (bossDB): A Cloud-Native Approach for Petascale Neuroscience Discovery. bioRxiv 2017;p. 217745.
21. Vathes LLC. Datajoint: A hub for developing, sharing, and publishing scientific data pipelines; 2018, https://datajoint.io.
22. Bock DD, Lee WCA, Kerlin AM, Andermann ML, Hood G, Wetzel AW, et al. Network anatomy and in vivo physiology of visual cortical neurons. Nature 2011;471(7337):177.
23. Kasthuri N, Hayworth K, Berger D, Schalek R, Conchello J, Knowles-Barley S, et al. Saturated Reconstruction of a Volume of Neocortex. Cell 2015 Jul;162(3):648–661.
24. Takemura Sy, Bharioke A, Lu Z, Nern A, Vitaladevuni S, Rivlin PK, et al. A visual motion detection circuit suggested by Drosophila connectomics. Nature 2013;500(7461):175.
25. Lee WCA, Bonin V, Reed M, Graham BJ, Hood G, Glattfelder K, 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, Schweighauser G, Hench J, Chicherova N, et al. Tomographic brain imaging with nucleolar detail and automatic cell counting. Scientific Reports 2016;6.
27. Busse M, Müller M, Kimm MA, Ferstl S, Allner S, Achterhold K, et al. Three-dimensional virtual histology enabled through cytoplasm-specific X-ray stain for microscopic and nanoscopic computed tomography. Proceedings of the National Academy of Sciences 2018;115(10):2293–2298.
28. Plaza S, Katz W, DVID; Retrieved June 2018. Https://github.com/janelia-flyem/dvid.
29. Lab S, cloud-volume; Retrieved June 2020. Https://github.com/seung-lab/cloud-volume.

30. Gürsoy D, De Carlo F, Xiao X, Jacobsen C. TomoPy: a framework for the analysis of synchrotron tomographic data. Journal of synchrotron radiation 2014;21(5):1188–1193.
31. Sommer C, Straehle C, Koethe U, Hamprecht FA. Ilastik: Interactive learning and segmentation toolkit. In: Biomedical Imaging: From Nano to Macro IEEE; 2011. p. 230–233.
32. Ronneberger O, Fischer P, Brox T. U-net: Convolutional networks for biomedical image segmentation. Springer; 2015. .
33. Ciresan D, Giusti A, Gambardella LM, Schmidhuber J. Deep neural networks segment neuronal membranes in electron microscopy images. In: Advances in neural information processing systems; 2012. p. 2843–2851.
34. Gray Roncal W, Pekala M, Kaynig-Fittkau V, Kleissas DM, Vogelstein JT, Pfister H, et al. VESICLE: volumetric evaluation of synaptic interfaces using computer vision at large scale. arXiv preprint 2014;.
35. Staffler B, Berning M, Boergens KM, Gour A, van der Smagt P, Helmstaedter M. SynEM, automated synapse detection for connectomics. eLife 2017;6.
36. Nunez-Iglesias J, Kennedy R, Plaza SM, Chakraborty A, Katz WT. Graph-based active learning of agglomeration (GALA): a Python library to segment 2D and 3D neuroimages. Frontiers in neuroinformatics 2014;8:34.
37. Nunez-Iglesias J, Kennedy R, Parag T, Shi J, Chklovskii DB. 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, Xenes D, Gray-Roncal WR. Transfer Learning Analysis of Image Processing Workflows for Electron Microscopy Datasets. In: 2019 53rd Asilomar Conference on Signals, Systems, and Computers IEEE; 2019. p. 1197–1201.
39. Bergstra J, Bengio Y. Random search for hyper-parameter optimization. Journal of Machine Learning Research 2012;13(Feb):281–305.
40. Jones DR, Schonlau M, Welch WJ. Efficient global optimization of expensive black-box functions. Journal of Global optimization 1998;13(4):455–492.
41. Azar MG, Dyer E, Kording K. Convex relaxation regression: Black-box optimization of smooth functions by learning their convex envelopes. arXiv preprint arXiv:160202191 2016;.
42. Prasad JA, Balwani AH, Johnson EC, Miano JD, Sampathkumar V, De Andrade V, et al. A three-dimensional thalamocortical dataset for characterizing brain heterogeneity. bioRxiv 2020;.
43. Gorgolewski K, Burns CD, Madison C, Clark D, Halchenko YO, Waskom ML, et al. Nipype: a flexible, lightweight and extensible neuroimaging data processing framework in python. Frontiers in neuroinformatics 2011;5:13.
44. Matelsky J, Substrate; Retrieved June 2018. Https://github.com/iscoe/substrate.
45. Google, Neuroglancer; Retrieved Sept 2018. Https://github.com/google/neuroglancer.
46. LaGrow TJ, Moore MG, Prasad JA, Davenport MA, Dyer EL. Approximating Cellular Densities from High-Resolution Neuroanatomical Imaging Data. In: Engineering in Medicine and Biology Society (EMBC); 2018. .
47. Kiar G, Gorgolewski KJ, Kleissas D, Roncal WG, Litt B, Wandell B, et al. Science In the Cloud (SIC): A use case in MRI Connectomics. Giga Science 2017;6(5):1–10.
48. Apache, Apache Hadoop; Retrieved June 2018. Https://hadoop.apache.org.
49. Dinov I, Van Horn J, Lozev K, Magsipoc R, Petrosyan P, Liu Z, et al. Efficient, distributed and interactive neuroimaging data analysis using the LONI pipeline. Frontiers in neuroinformatics 2009;3:22.
50. Afgan E, Baker D, Van den Beek M, Blankenberg D, Bouvier D, Čech M, et al. The Galaxy platform for accessible, reproducible and collaborative biomedical analyses: 2016 update. Nucleic acids research 2016;44(W1):W3–W10.
51. Berthold MR, Cebron N, Dill F, Gabriel TR, Kötter T, Meinl T, et al. KNIME-the Konstanz information miner: version 2.0 and beyond. AcM SIGKDD explorations Newsletter 2009;11(1):26–31.
52. Glatard T, Da Silva RF, Boujelben N, Adalat R, Beck N, Rioux P, et al. Boutiques: an application-sharing system based on Linux containers. Neuroinformatics 2015;.
53. Labs C, DRAY: Docker Workflow Engine; Retrieved June 2018. Http://www.dray.it.
54. Air-tasks; Retrieved Sept 2018. Https://github.com/wongwill86/air-tasks.
55. Matelsky J, Intern: Integrated Toolkit for Extensible and Reproducible Neuroscience;. Https://github.com/jhuapl-boss/intern.
56. Matelsky JK, Rodriguez L, Xenes D, Gion T, Hider R, Wester B, et al. Intern: Integrated toolkit for extensible and reproducible neuroscience. bioRxiv 2020;.

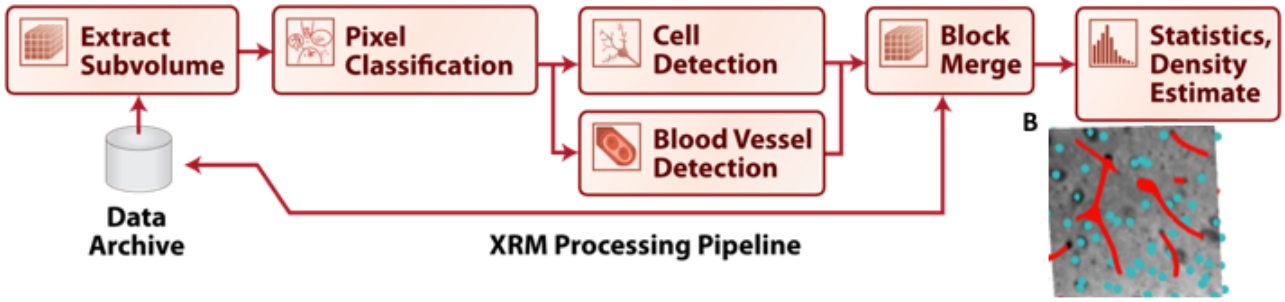


Figure 1. Workflow for processing XRM data to produce cell and vessel location estimates. Raw pixels are used to predict probabilities of boundaries, followed by detection of cell bodies and blood vessels. Finally, cell density estimates are created. Panel A shows the reconstruction pipeline, whereas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

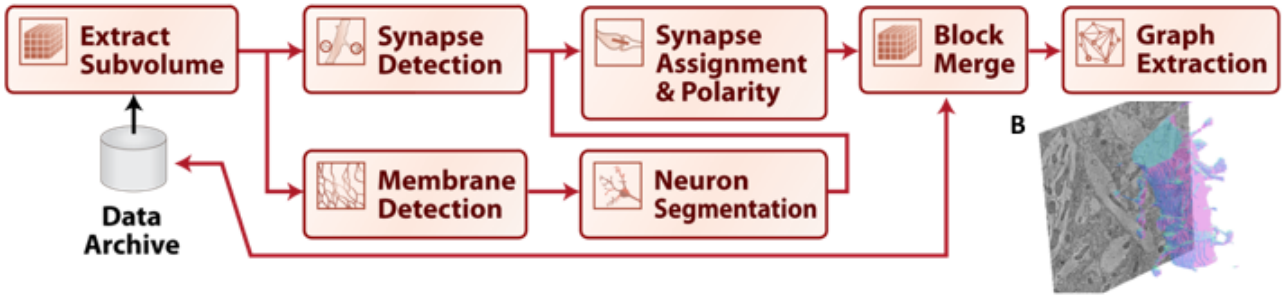


Figure 2. Canonical Workflow for Graph Estimation in EM data volumes. This workflow provides the ability to reconstruct a nanoscale map of brain circuitry at the single synapse level. The procedure of mapping raw image stacks to graphs representing synapse-level connectomes consists of synapse and membrane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

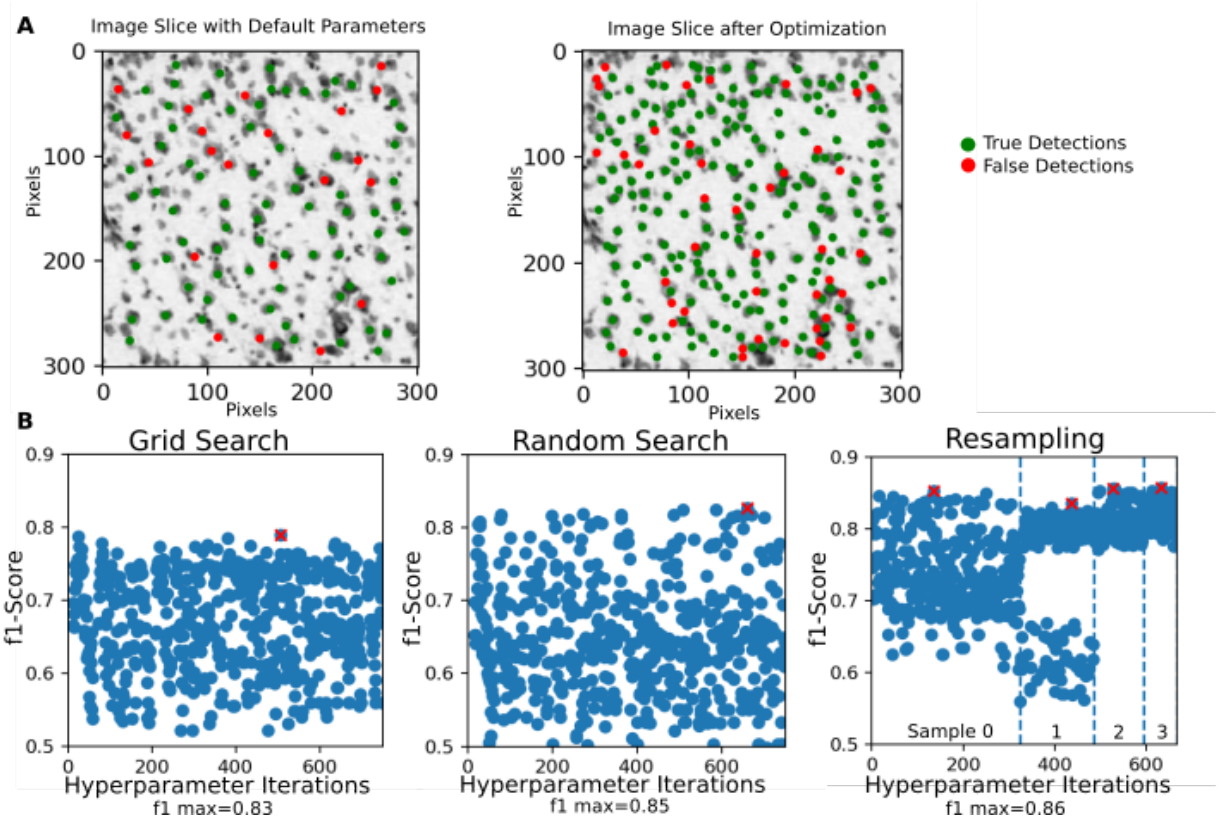


Figure 3. Use case of optimizing a pipeline for light microscopy data, comparing grid search, random search, and the random resampling approach described in the text. We demonstrate these tools on a light microscopy dataset, leveraging methods originally developed for XRM – showcasing the potential for 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 'x'. Automating this process using SABER allows for rapid deployment and optimization.

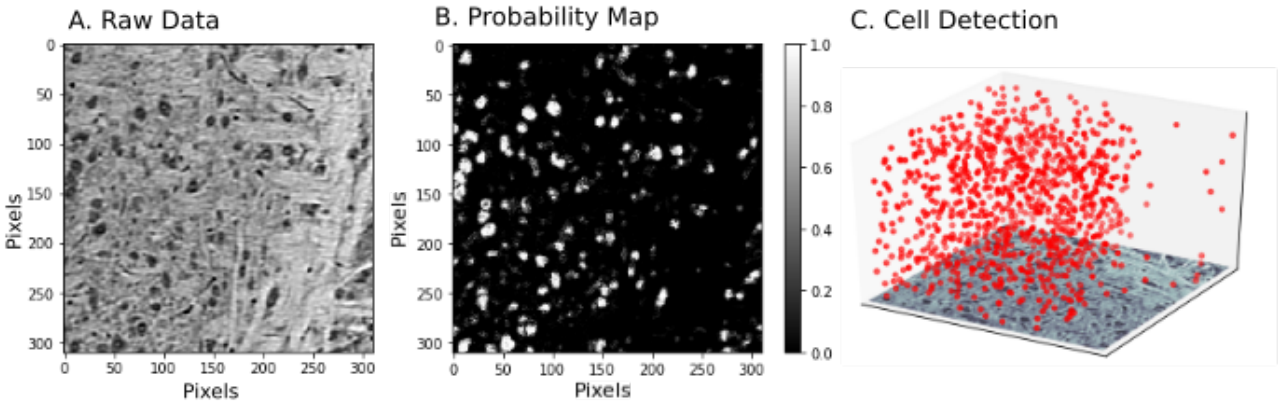


Figure 4. Example deployment of pipeline over spatial dataset, in this case cell detection in XRM data. An example slice of raw data can be seen in Panel A. The pipeline in Fig.ĩ was used to classify pixels (Panel B) and detect cells. From the cells, a three dimensional scatter plot of the positions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

	SABER/ CONDUIT	**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	N/A
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	N/A
Parallel Processing Model	Block-merge	None	None	None	Block-merge	N/A
Workflow deployment for neuroimaging	Yes	No	No	No	Yes	N/A
Workflow optimization for neuroimaging	Yes	No	No	No	No	N/A
Tool Benchmarking and Datasets	Yes	No	No	No	No	N/A
EM tool library	Yes	No	No	No	Yes	N/A
Tools for Other Modalities	Yes	No	No	No	No	N/A

Table 1. Comparison of existing projects related to workflow execution of neuroimaging pipelines with lighter cells highlighting desirable features, medium cells highlight partial implementations of desirable features, and darker cells highlighting limitations. 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 less flexibility for users. Similar projects such as TOIL, Galaxy, and CWL-Airflow lack neuroimaging specific features to enable the use cases described in Section 3. 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 1) interfacing with existing solutions, 2) providing a library of portable, Dockerized neuroimaging tools, and 3) providing scripting to analyze large-scale neuroimaging datasets.

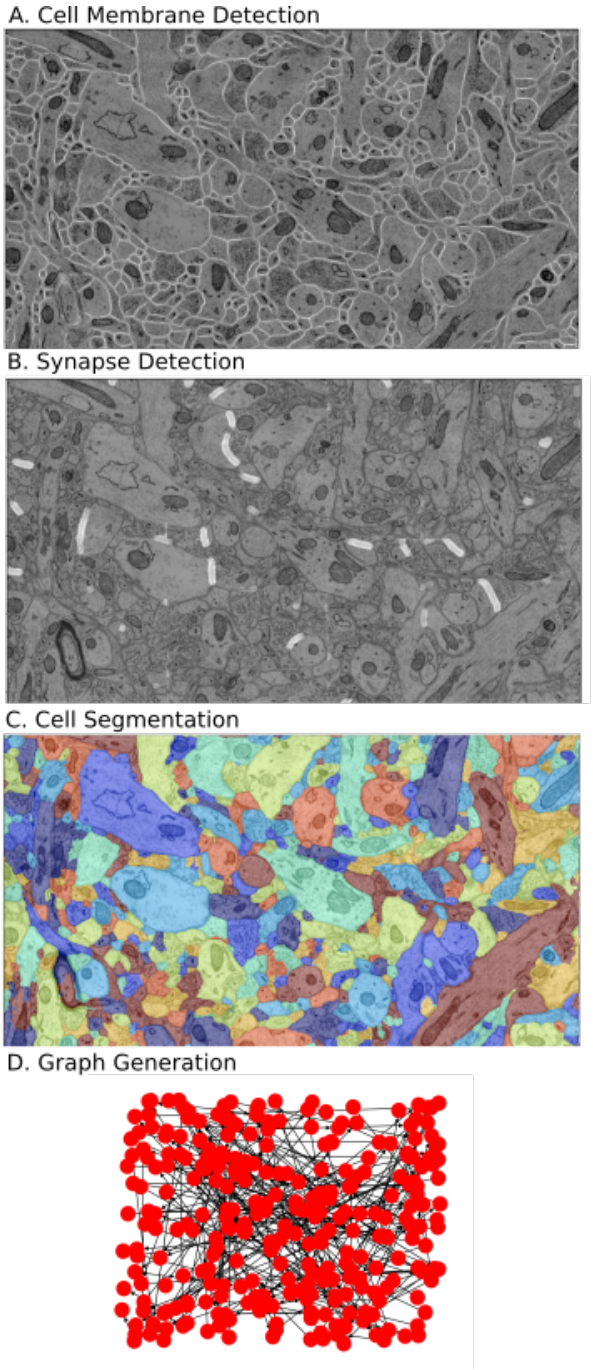


Figure 5. Example deployment of EM segmentation pipeline to extract graphical models of connectivity from raw images. The processing pipeline (Fig.1) consists of neural network tools to perform A) membrane detection and B) synapse detection. This is followed by a segmentation tool (Panel C). Finally, segmentation and synapses are associated to create a graphical model. Visualizations of segmentations are done with Neuroglancer [45], a tool compatible with SABER and integrated with the bossDB [20] system.

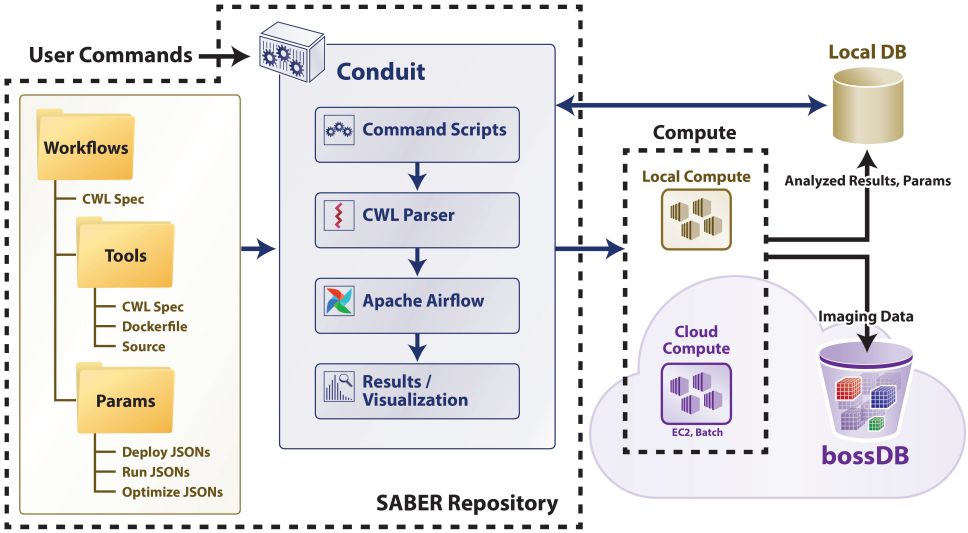


Figure 6. The architecture and components of SABER. Tools, workflows, and parameters for individual use cases (optimization, deployment) are captured in a file structure using standardized CWL specifications and configuration files. The core of the framework (called CONDUIT) is run locally in a Docker 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.

Fig 1

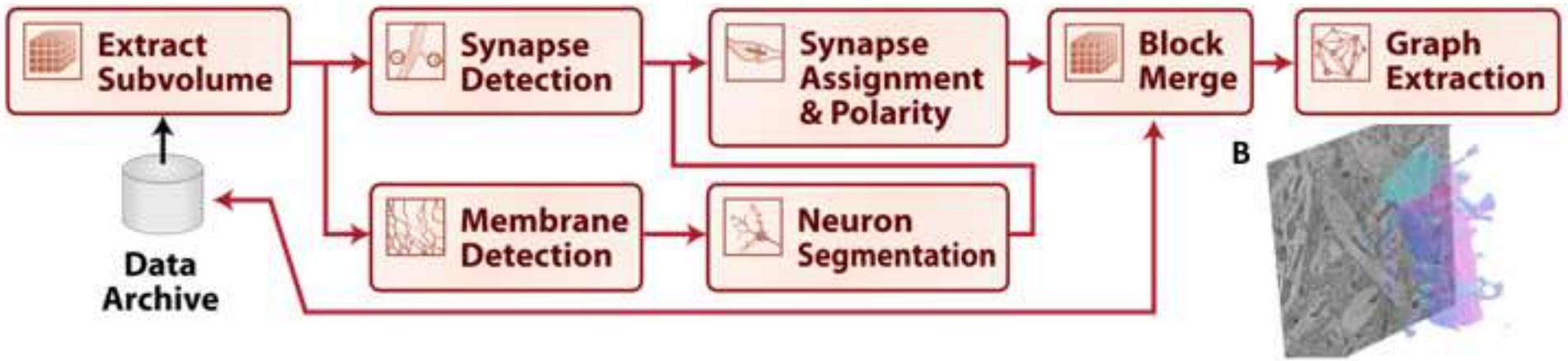
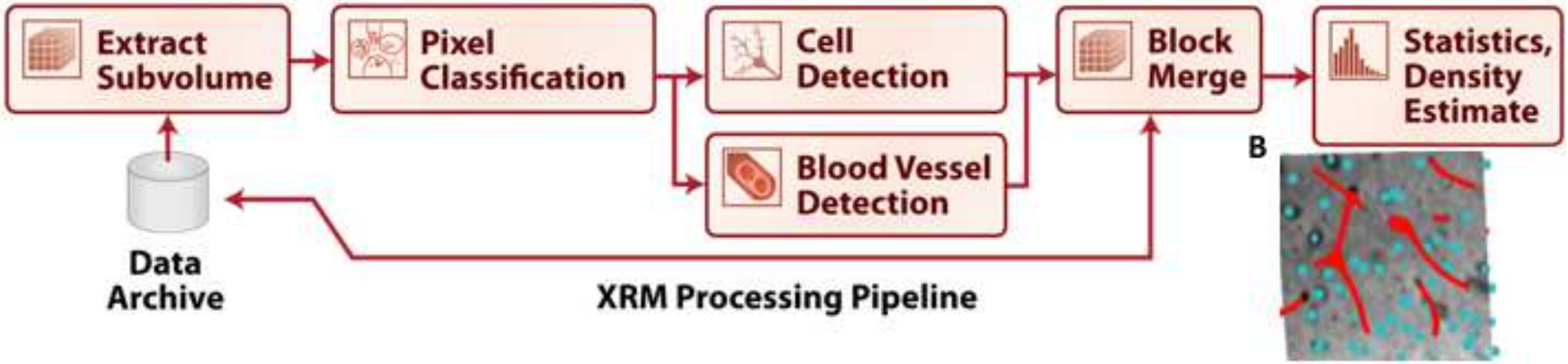


Fig 2



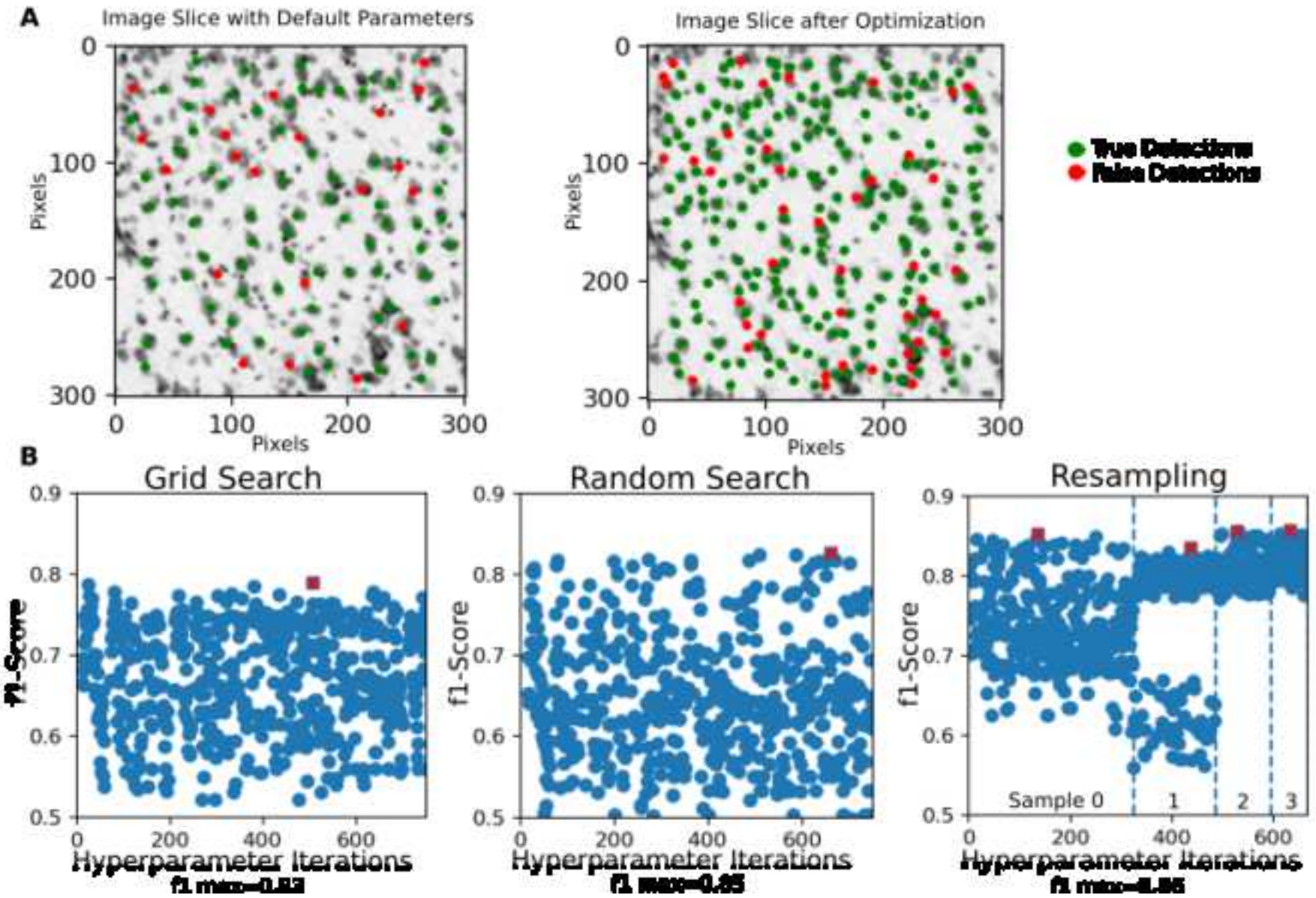


Fig 4

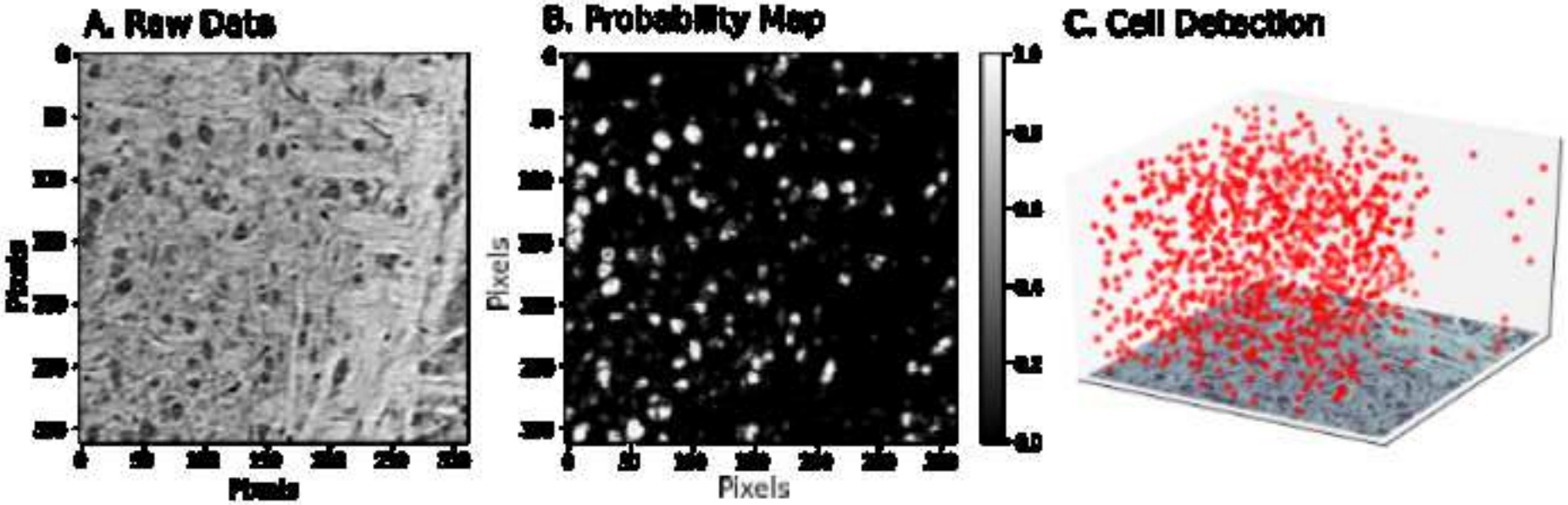


Fig 5

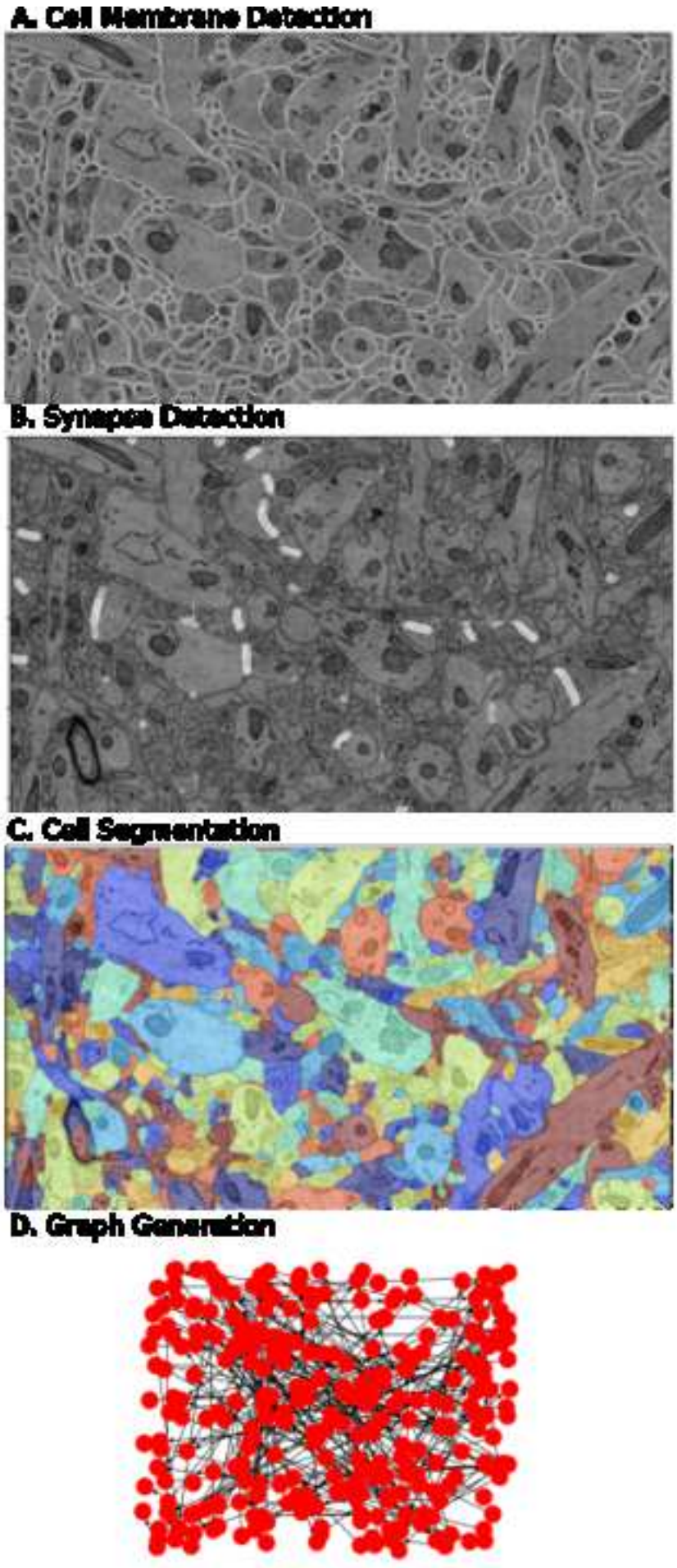
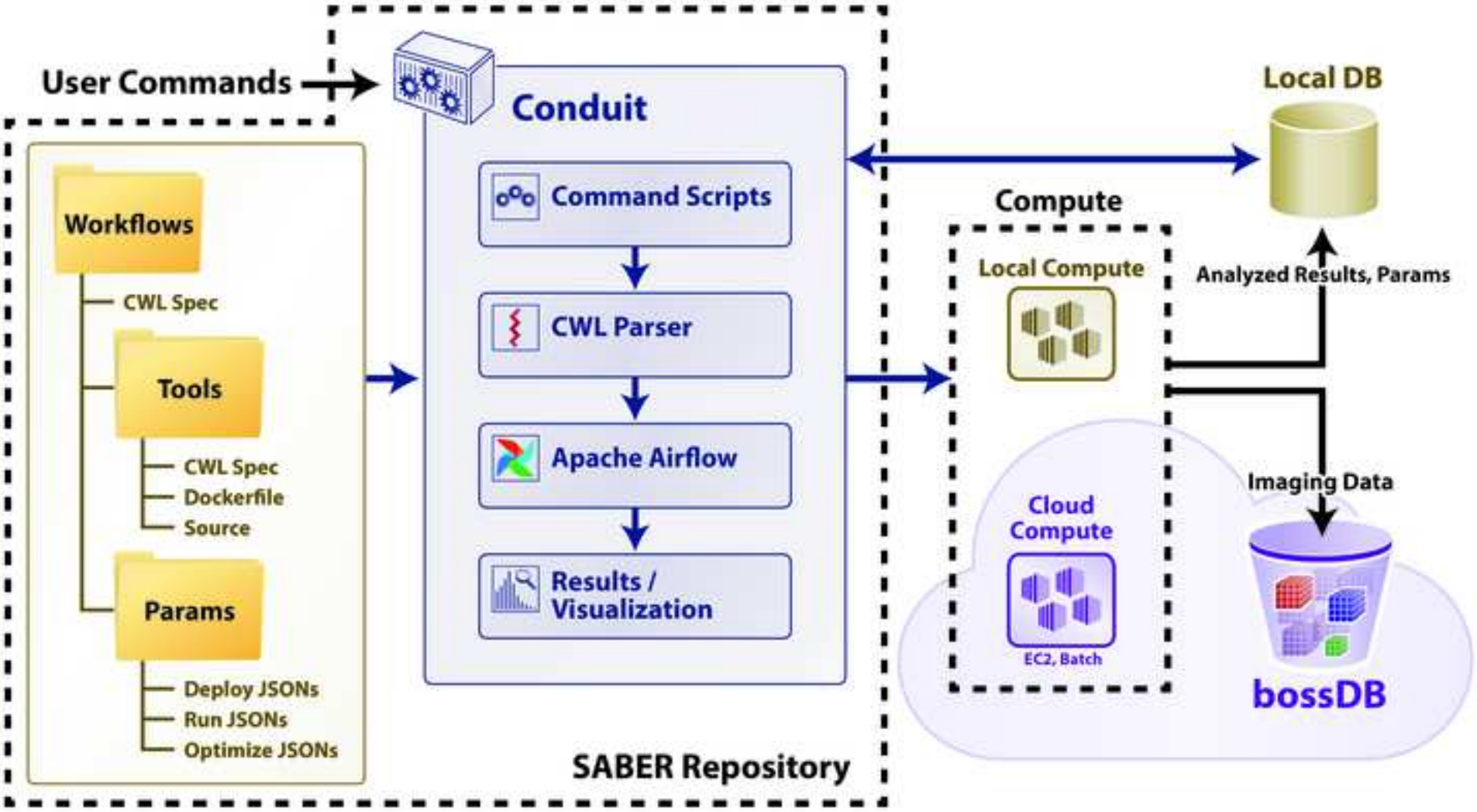


Fig 6



Red-text for revisions