Neuron

NeuroResource

Neuroscience Cloud Analysis As a Service: An open-source platform for scalable, reproducible data analysis

Highlights

- The NeuroCAAS platform provides reproducible data analysis infrastructure at scale
- Reproducible, cloud-based infrastructure enables novel analysis design
- Popular data analyses adapted to NeuroCAAS are faster and cheaper than alternatives

Authors

Taiga Abe, Ian Kinsella,
Shreya Saxena, ..., John Zhou,
Liam Paninski, John P. Cunningham

Correspondence

jpc2181@columbia.edu

In brief

Computing infrastructure is a fundamental part of neural data analysis. Abe et al. present an open-source, cloud-based platform called NeuroCAAS to automatically build reproducible computing infrastructure for neural data analysis. They show that NeuroCAAS supports novel analysis design and can improve the efficiency of popular existing methods.

Abe et al., 2022, Neuron *110*, 2771–2789
September 7, 2022 © 2022 Elsevier Inc.
https://doi.org/10.1016/j.neuron.2022.06.018

CellPress

Neuron

CellPress

NeuroResource

Neuroscience Cloud Analysis As a Service: An open-source platform for scalable, reproducible data analysis

Taiga Abe,[1,2,4] Ian Kinsella,[1,2,5] Shreya Saxena,[1,2,3,5,8] E. Kelly Buchanan,[1,2,4] Joao Couto,[7] John Briggs,[1] Sian Lee Kitt,[6] Ryan Glassman,[6] John Zhou,[6] Liam Paninski,[1,2,3,4,5] and John P. Cunningham[1,2,3,5,9,*]

[1] Mortimer B. Zuckerman Mind Brain Behavior Institute, Department of Neuroscience, Columbia University, New York, NY 10027, USA
[2] Center for Theoretical Neuroscience, Columbia University, New York, NY 10027, USA
[3] Grossman Center for the Statistics of Mind, Columbia University, New York, NY 10027, USA
[4] Department of Neuroscience, Columbia University Medical Center, Columbia University, New York, NY 10027, USA
[5] Department of Statistics, Columbia University, New York, NY 10027, USA
[6] Department of Computer Science, Columbia University, New York, NY 10027, USA
[7] Department of Neurobiology, David Geffen School of Medicine, University of California, Los Angeles, CA 90095, USA
[8] Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL 32607, USA
[9] Lead contact
*Correspondence: jpc2181@columbia.edu

https://doi.org/10.1016/j.neuron.2022.06.018

SUMMARY

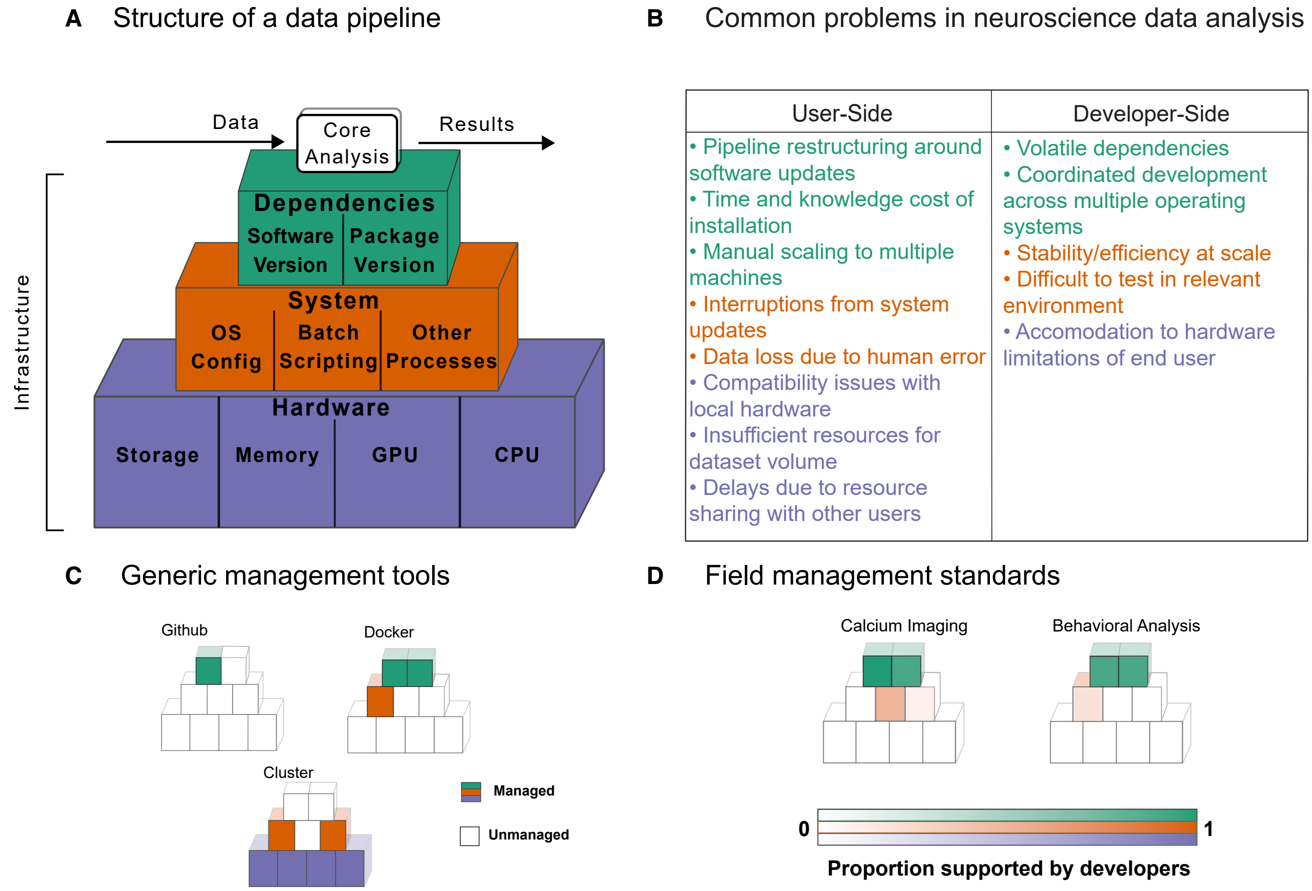
A key aspect of neuroscience research is the development of powerful, general-purpose data analyses that process large datasets. Unfortunately, modern data analyses have a hidden dependence upon complex computing infrastructure (e.g., software and hardware), which acts as an unaddressed deterrent to analysis users. Although existing analyses are increasingly shared as open-source software, the infrastructure and knowledge needed to deploy these analyses efficiently still pose significant barriers to use. In this work, we develop Neuroscience Cloud Analysis As a Service (NeuroCAAS): a fully automated open-source analysis platform offering automatic infrastructure reproducibility for any data analysis. We show how NeuroCAAS supports the design of simpler, more powerful data analyses and that many popular data analysis tools offered through NeuroCAAS outperform counterparts on typical infrastructure. Pairing rigorous infrastructure management with cloud resources, NeuroCAAS dramatically accelerates the dissemination and use of new data analyses for neuroscientific discovery.

INTRODUCTION

Driven by the constant evolution of new recording technologies and the vast quantities of data that they generate, neural data analysis—which aims to build the path from these datasets to scientific understanding—has grown into a centrally important part of modern neuroscience, enabling significant new insights into the relationships between neural activity, behavior, and the external environment (Paninski and Cunningham, 2018). Accompanying this growth, however, neural data analyses have become much more complex. Historically, the software implementation of a data analysis (what we call the core analysis; Figure 1A) was typically a small, isolated code script with few dependencies. In stark contrast, modern core analyses routinely incorporate video-processing algorithms (Pnevmatikakis et al., 2016; Pachitariu et al., 2017; Mathis et al., 2018; Zhou et al., 2018; Giovannucci et al., 2019), deep neural networks (Batty et al., 2016; Gao et al., 2016; Lee et al., 2017; Parthasarathy et al., 2017; Mathis et al., 2018; Pandarinath et al., 2018; Giovannucci et al., 2019), sophisticated graphical models (Yu et al., 2009; Wiltschko et al., 2015; Gao et al., 2016; Wu et al., 2020), and other cutting-edge machine learning methods (Pachitariu et al., 2016; Lee et al., 2017) to create general-purpose tools applicable to many datasets.

To support this increasing complexity, core analysis software is increasingly coupled to underlying analysis *infrastructure* (Figure 1A): software dependencies like the deep learning libraries PyTorch and TensorFlow (Abadi et al., 2016; Paszke et al., 2019), system-level dependencies to manage jobs and computing resources (Merkel, 2014), and hardware dependencies such as a precisely configured central processing unit (CPU), access to a graphics processing unit (GPU), or a required amount of device memory. Figure 1A shows how these individual components form a full infrastructure *stack*: the necessary but largely ignored foundation of resources enabling all data analyses (Demchenko et al., 2013; Jararweh et al., 2016; Zhou et al., 2016).

Neglected infrastructure has immediate implications already familiar to the neuroscience community: for every novel analysis,



User-Side	Developer-Side
• Pipeline restructuring around software updates • Time and knowledge cost of installation • Manual scaling to multiple machines • Interruptions from system updates • Data loss due to human error • Compatibility issues with local hardware • Insufficient resources for dataset volume • Delays due to resource sharing with other users	• Volatile dependencies • Coordinated development across multiple operating systems • Stability/efficiency at scale • Difficult to test in relevant environment • Accomodation to hardware limitations of end user

Figure 1. Data analysis infrastructure

(A) Core analysis code depends upon an infrastructure stack.

(B) Common problems arise at each layer of this infrastructure stack for analysis users and developers.

(C) Many common management tools deal only with one or two layers in the infrastructure stack, leaving gaps that users and developers must fill manually.

(D) In common neural data analysis tools for calcium imaging and behavioral analysis many infrastructure components are not managed by analysis developers and implicitly delegated to the user (see STAR Methods for full details and supporting data in Tables S2 and S1).

analysis users must spend labor and financial resources on hardware setup, software troubleshooting, unexpected interruptions during long analysis runs, processing constraints due to limited "on-premises" computational resources, and more (Figure 1B). However, far from simply being a nuisance, neglected infrastructure has wide reaching and urgent scientific consequences. Most prominently, infrastructure impacts analysis reproducibility. As data analyses become more dependent on complex infrastructure stacks, it becomes extremely difficult for analysis developers to work reproducibly (Monajemi et al., 2019; Nowogrodzki, 2019). The current treatment of analysis infrastructure is a major contributor to the endemic lack of reproducibility suffered by modern data analysis (Crook et al., 2013; Hinsen, 2015; Stodden et al., 2018; Krafczyk et al., 2019; Raff, 2019), and infrastructure-based barriers have been noted to impede the proliferation of new neuroscientific tools (Magland et al., 2020). Specific cases where seemingly small infrastructure issues directly affect the representation of data-derived quantities have been documented across the biological sciences (Ghosh et al., 2017; Miller, 2006; Glatard et al., 2015). Analogously, for machine learning methods, infrastructure components can dictate model performance (Sculley et al., 2015; Radiuk, 2017), and a recent survey of this literature observed that although local compute clusters claim to address the issue of hardware availability, *none* of the studies that required use of a compute cluster were reproducible (Raff, 2019).

Major efforts have been made by journals (Donoho, 2010; Hanson et al., 2011; Editorial, 2014) and funding agencies (Carver et al., 2018) to encourage the sharing of core analysis software. Additionally, new tools have been developed to address related neuroscientific challenges like the formatting (Teeters et al., 2015; Rübel et al., 2019, 2021) and storage of data (Dandi Team, 2019), or workflow management on existing infrastructure (Yatsenko et al., 2015; Gorgolewski et al., 2011) (see discussion for a detailed overview). However, these important efforts still neglect key issues in the configuration of infrastructure stacks. Despite calls to improve standards of practice in the field (Neuro Cloud Consortium, 2016) and work in fields such as astronomy, genomics, and high energy physics (Hoffa et al., 2008; Riley, 2010; Goecks et al., 2010; Zhou et al., 2016;

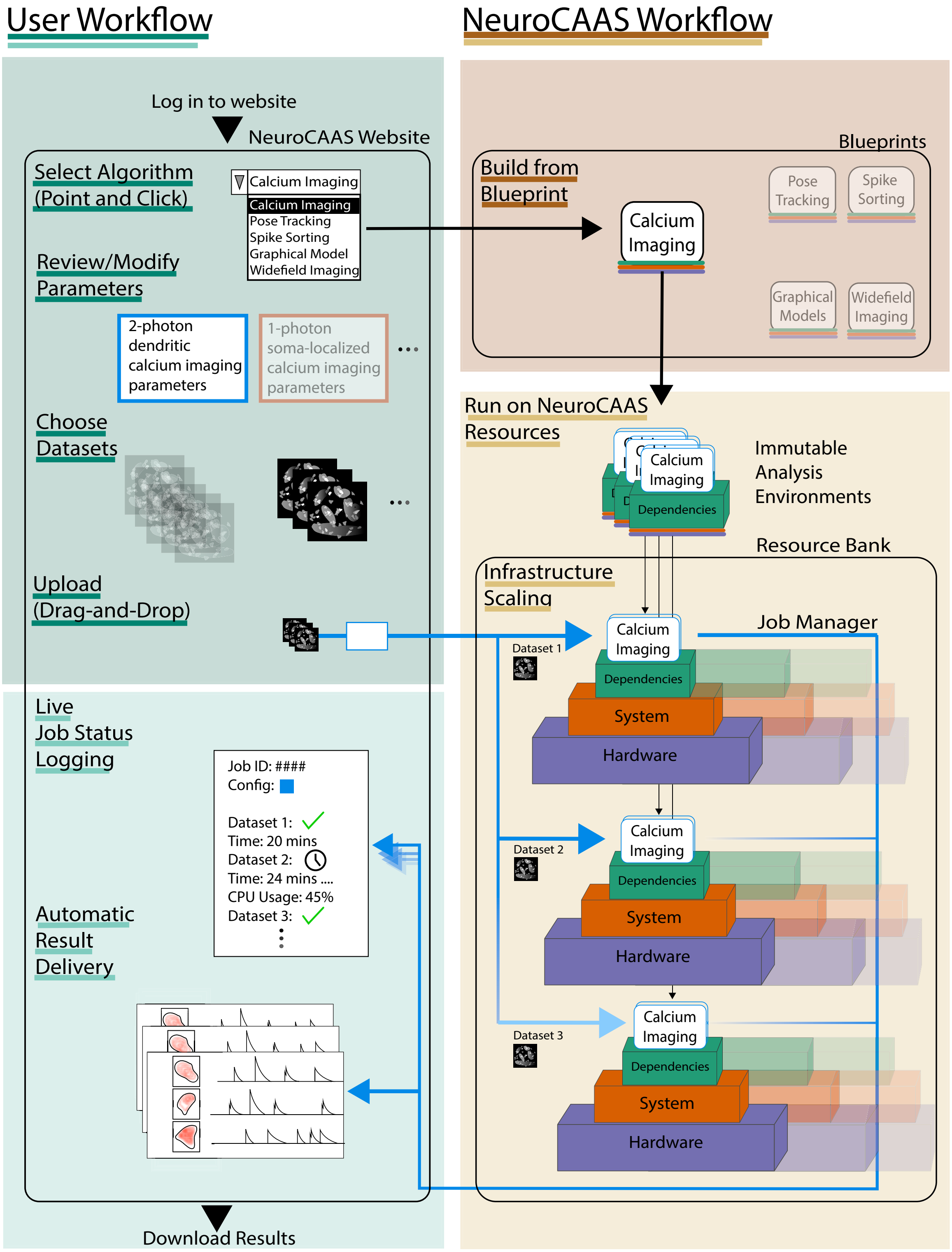


Figure 2. Overview of NeuroCAAS user workflow

Left indicates the user's experience; right indicates the work that NeuroCAAS performs. The user chooses from the analyses encoded in NeuroCAAS. They then modify corresponding configuration parameters as needed. Finally, the user uploads dataset(s) and a configuration file for analysis. NeuroCAAS detects upload events and deploys the requested analysis using an infrastructure blueprint (blueprints for instant reproducibility). NeuroCAAS builds the appropriate number of

(legend continued on next page)

Chen et al., 2019; Monajemi et al., 2019), there has been little concrete progress in our field toward a scientifically acceptable infrastructure solution for many popular core analyses. Some tools—compute clusters, versioning tools like Github (https://github.com), and containerization services like Docker (Merkel, 2014)—provide various infrastructure components (Figure 1C), but it is nontrivial to combine these components into a complete infrastructure stack. The ultimate effect of these partially used toolsets is a hodgepodge of often slipshod infrastructure practices (Figure 1D; supplemental information in Tables S1 and S2).

Critically, management of these issues most often falls upon trainees who are neither scientifically rewarded (Landhuis, 2017; Chan Zuckerberg Initiative, 2019) nor specifically instructed (Merali, 2010) on how to set up increasingly complex core analyses and infrastructure stacks, reliant on whatever resources are available on hand. We term this conventional model infrastructure-as-graduate-student, or IaGS—infrastructure stacks treated as a scientific afterthought, delegated entirely to underresourced trainees and operating as a silent source of errors and inefficiency. The IaGS status quo fails any reasonable standard of scientific rigor, reduces the accessibility of valuable analytical tools, and impedes scientific training and progress.

Of course, infrastructure challenges are not specific to neuroscience, or even science generally. Entities that deploy software at industrial scale have recently adopted the infrastructure-as-code (IaC) paradigm, automating the creation and management of infrastructure stacks (Morris, 2016; Aguiar et al., 2018). In contrast to IaGS approaches, the IaC paradigm begins with a code document that completely specifies the infrastructure stack supporting any given core software. From this code document, the corresponding infrastructure stack can be assembled automatically (most often on a cloud platform), in a process called deployment. After deployment, anyone with access to the platform can use the core software in question without knowledge of its underlying infrastructure stackwhile still having the assurance that core software is functioning exactly as indicated in the corresponding code document. Altogether, IaC enables reproducible usage at scale, skirting all of the issues shown in Figure 1B. Despite these benefits, there has been no previous effort to extend IaC to general-purpose neuroscience data analyses and associated infrastructure stacks.

In response, we developed Neuroscience Cloud Analysis As a Service (NeuroCAAS), an IaC platform that pairs core analyses for neuroscience data with bespoke infrastructure stacks through deployable code documents. NeuroCAAS assigns each core analysis a corresponding infrastructure stack, using a set of modular components concisely specified in code (see section NeuroCAAS builds complete infrastructure stacks for details). NeuroCAAS stores the specification of this core analysis and infrastructure stack in a code document called a blueprint, which any analysis user can then deploy to analyze their data. To maximize the scale and accessibility benefits of our platform, we provide an open-source web interface to NeuroCAAS (see section NeuroCAAS supports simple use and development), available to the neuroscience community at large. The result is scalable, reproducible, drag-and-drop usage of neural data analysis: neuroscientists can log on to the NeuroCAAS website, set some parameters for an analysis, and simply submit their data. A new infrastructure stack is then deployed on the cloud according to a specified blueprint and autonomously produces analysis results, which are returned to the user. This aspect of NeuroCAAS warrants emphasis, as it diverges starkly from traditional scientific practice: NeuroCAAS is not only a platform design or suggestion that the reader can attempt to recreate on their own; instead, NeuroCAAS is offered as an open-source infrastructure platform available for immediate use, via a website (www.neurocaas.org).

Our results sections are organized as follows: We first describe IaC analysis infrastructure on NeuroCAAS (NeuroCAAS builds complete infrastructure stacks) and how it addresses common engineering challenges related to analysis reproducibility, accessibility, and scale (NeuroCAAS supports simple use and development). In existing platforms leave infrastructure gaps, we compare NeuroCAAS's solution to these engineering challenges with features of existing data analysis platforms. Next, in sections NeuroCAAS simplifies large data pipelines: widefield imaging protocol and NeuroCAAS stabilizes deep learning models: ensemble markerless tracking, we show how NeuroCAAS can enable novel analyses designed to take advantage of the platform's infrastructure benefits. Finally, in NeuroCAAS is faster and cheaper than IaGS analogs, we quantify the performance of popular data analyses on NeuroCAAS and find that analyses encoded in blueprints are cheaper and faster than analogs run on local infrastructure (e.g., a compute cluster).

RESULTS

NeuroCAAS's primary technical contribution is a method to precisely specify the entire infrastructure stack underlying any core analysis and reproduce it on demand. Treating core analysis and infrastructure as a unified whole within NeuroCAAS makes analyses more reproducible and accessible at scale than existing alternatives.

In the simplest use case, users simply log in to the platform and drag-and-drop their dataset(s) into a web browser (Figure 2, top left), sending it to cloud-based user storage. After specifying a set of developer-defined parameters to apply to the selected dataset(s), they can submit a NeuroCAAS "job." No further user input is needed: given the relevant datasets(s) and parameters, NeuroCAAS sets up core analysis for each dataset on an entirely new infrastructure stack from the corresponding blueprint (Figure 2, black arrows). NeuroCAAS then pulls data and parameters independently into each infrastructure stack (Figure 2, blue arrows), providing scalable and reproducible computational processing as needed (Figure 2, bottom right). Analysis outputs (including live status logs and a complete record of the

IAEs (immutable analysis environments for software infrastructure) and corresponding hardware instances (immutable analysis environments for software infrastructure). Multiple infrastructure stacks may be deployed in parallel for multiple datasets and the job manager (job managers for system infrastructure) automatically handles input and output scaling. The deployed resources persist only as necessary, and results, as well as diagnostic information, are automatically routed back to the user. See Figure S1 for comparison with IaGS, and Figure S3 for IAE list.

job's inputs and infrastructure) are then delivered back to timestamped folders in user storage for inspection by analysis users (Figure 2, bottom left), and finally, infrastructure stacks are dissolved when data processing is complete (Figure 2, bottom right). As an example, Video S1 shows how users can train three separate DeepGraphPose models (Wu et al., 2020) on three separate datasets simultaneously using the NeuroCAAS web interface.

NeuroCAAS builds complete infrastructure stacks

The structure of NeuroCAAS naturally solves the issues of reproducibility, accessibility, and scale that burden existing infrastructure tools and platforms. NeuroCAAS partitions a complete infrastructure stack into three decoupled parts that together are sufficient to support virtually any given core analysis. First, to address all software level infrastructure, NeuroCAAS offers all analyses in immutable analysis environments or IAEs (immutable analysis environments for software infrastructure). Second, to address system configuration, each NeuroCAAS analysis has a built-in job manager (job managers for system infrastructure) that automates all of the logistical tasks associated with analyzing data: configuring hardware, logging outputs, parallelizing jobs, and more. Third, to provide reproducible computing hardware on demand, NeuroCAAS manages a resource bank (resource banks for hardware infrastructure) built on the public cloud, making the service globally accessible at unmatched scale. For a given core analysis, the configuration of these three infrastructure components is concisely summarized in a NeuroCAAS blueprint, from which they can be automatically rebuilt (blueprints for instant reproducibility). We describe component implementation in further depth in method details.

Immutable analysis environments for software infrastructure

On NeuroCAAS, all core analyses run inside IAEs. An IAE is an isolated computing environment containing the installed core analysis code and all necessary software dependencies, similar to a Docker container (Merkel, 2014). Importantly, an IAE also contains a single script that parses input and parameters in a prescribed way and runs the steps of core analysis workflow (Figure 2, right; Figure S6, top left). The fact that analysis workflow is entirely governed by this script (i.e., non-interactive) makes our analysis environments immutable. IAEs eliminate the possibility of bugs resulting from incompatible dependencies, mid-analysis misconfiguration (Figure S1A, installation and troubleshooting), or other so-called "user degrees of freedom" and ensure that analyses are run within *developer*-defined workflows. Immutability has a long history as a principle of effective programming and resource management in computer science (Bloch, 2008; Morris, 2016), and in this context, it is closely related to the view that data analysis should be automated as much as possible (Tukey, 1962; Waltz and Buchanan, 2009). These views are justified by observed benefits to analysis at scale, which we leverage in sections NeuroCAAS simplifies large data pipelines: widefield imaging protocol and NeuroCAAS stabilizes deep learning models: ensemble markerless tracking.

Each IAE has a unique ID, and analysis updates can be recorded in IAEs linked through blueprint versions (see blueprints for instant reproducibility for details). We have currently implemented 22 analyses in a series of IAEs (Table S3) and are actively developing more (see www.neurocaas.org for current options).

Job managers for system infrastructure

Given a dataset and analysis parameters, how does NeuroCAAS setup the right IAE and computing hardware to process these inputs? This configuration is the responsibility of the NeuroCAAS job manager, which monitors analysis progress and returns timestamped job outputs to user storage from the IAE (including live job logs). Although similar in these regards to a cluster workload manager like slurm (Yoo et al., 2003) (Figure 2, blue arrows), the NeuroCAAS job manager does not assign jobs to running infrastructure, but rather sets up all other infrastructure components "on the fly," removing the need for manual infrastructure maintenance (Figure 2; black arrows). The job manager for each analysis functions according to a code "protocol" that describes what steps should be taken when a new NeuroCAAS job is requested. Importantly, protocols can be customized for each analysis, allowing developers to implement simple features like input parsing, or complex multi-stack workflows as described in sections NeuroCAAS simplifies large data pipelines: widefield imaging protocol and NeuroCAAS stabilizes deep learning models: ensemble markerless tracking.

Resource banks for hardware infrastructure

To automatically reproduce infrastructure on demand, we crucially need a way to create identical hardware configurations across multiple users of the same analysis, who may be analyzing data simultaneously at many different locations around the world. This key requirement is handled by the NeuroCAAS resource bank. The NeuroCAAS resource bank can make hardware available through pre-specified instances: bundled collections of virtual CPUs, memory, and GPUs that can emulate any number of familiar hardware configurations (e.g., personal laptop, workstation, on premise cluster). However, unlike these persistent computing resources, the NeuroCAAS resource bank is built upon globally available, virtualized compute hardware offered through the public cloud (currently Amazon Web Services [AWS]). At any time, the resource bank can provide a large number of effectively identical hardware instances to execute a particular task (Figure 2, bottom right). The reproducible nature of hardware instances in the resource bank complements the immutability of workflows imposed by IAEs. By default, we fix a single instance type per analysis in order to facilitate reproducibility (see method details for more information).

Blueprints for instant reproducibility

For any given analysis, each of NeuroCAAS′s infrastructure components has a specification in code (IAEs and resource bank instances have IDs, job managers have protocol scripts). The collection of all infrastructure identifying code associated with a given NeuroCAAS analysis is stored in the blueprint of that analysis (Figure 2, top right—for an example, see Figure S2), from which new copies of the infrastructure stack can be deployed at will, providing reproducibility by design. Despite sustained efforts to promote reproducible research, (Buckheit and Donoho, 1995), in many typical cases, data analysis remains frustratingly non-reproducible (Crook et al., 2013; Gorgolewski et al., 2017; Stodden et al., 2018; Raff, 2019). NeuroCAAS sidesteps all of the typical barriers to reproducible research by tightly coupling the creation and function of infrastructure stacks to their documentation. For transparency, NeuroCAAS stores all currently available and developing analysis blueprints in a public code repository (see method details). Updates made to any component of an infrastructure stack on NeuroCAAS (IAEs, job managers, or hardware instances) can *only* be implemented through subsequent deployments of updated blueprints. We record these changes systematically using simple version control on the blueprint itself, ensuring a publicly visible record of analysis development.

NeuroCAAS supports simple use and development

Users analyze data on NeuroCAAS solely through interactions with cloud storage. Therefore, NeuroCAAS supports any interface that allows users to transfer data files to and from cloud storage. The standard interface to NeuroCAAS is a website, www.neurocaas.org, where users can sign up for an account, browse analyses, deposit data, and monitor analysis progress until results are returned to them as described in Figure 2. We will describe other interfaces to NeuroCAAS in sections NeuroCAAS simplifies large data pipelines: widefield imaging protocol and discussion. Regardless of interface, there is no need to manage persistent compute resources during or after analysis, and costs directly reflect usage time.

For comparison, IaGS begins with a number of time-consuming manual steps, including hardware acquisition, hardware setup, and software installation. With a functional infrastructure stack in hand, the user must prepare datasets for analysis, manually recording analysis parameters and monitoring the system for errors as they work. Although parallel processing is possible, it must be scripted by the user, and in many cases, datasets are run serially. What results from IaGS is massive inefficiency of time and resources. Users must also support the cost of new hardware ‘‘up front,’’ before ever seeing the scientific value of the infrastructure that they are purchasing. Likewise, labs or institutions must pay support costs to maintain infrastructure when it is not being used and replace components when they fail or become obsolete (see Figure S1 for a side-by-side comparison with NeuroCAAS). Two editorial remarks bear mentioning at this point: first, the stark difference laid out in these workflows is the essence of IaGS versus IaC and explains the dominance of IaC in modern industrial settings. Second, NeuroCAAS is and will remain an open-source tool for the scientific community, in keeping with its sole purpose of improving the reproducibility and dissemination of neuroscience analysis tools.

For analysis developers interested in NeuroCAAS, we designed a developer workflow and companion python package that streamlines the process of migrating an existing analysis to the NeuroCAAS platform (see method details and Figure S6 for an overview). Our developer workflow abstracts away the cloud infrastructure that NeuroCAAS is built on, allowing for analysis development entirely from the developer’s command line. We highlight several key features of this workflow here. Importantly, we do not expect any previous experience in containerization technology, cloud tools, or IaC from developers.

Curated deployments

After setting up an IAE and initializing a corresponding blueprint, developers submit blueprints and test data to our code repository in publicly visible pull requests. NeuroCAAS team members then review the submitted blueprint and deploy the corresponding infrastructure stack in ‘‘test mode,’’ reviewing outputs, requesting updates, and ensuring stated function in a public forum before releasing the analysis to users.

Improving analysis robustness

A central challenge of building general-purpose data analysis is the difficulty of anticipating all analysis use cases as a developer and ensure robustness to all possible datasets. This is true even with the testing, review and development practices that NeuroCAAS puts in place. However, NeuroCAAS′s design has several features intended to accelerate the process of improving analysis robustness both during and after initial deployment. When errors occur, users can refer the analysis developer to version controlled analysis outputs using standard interfaces like Github issues, greatly simplifying error replication. Developers can then set up fixes on the same infrastructure and update the analysis blueprint in subsequent deployments. Since infrastructure is rebuilt from the blueprint for each NeuroCAAS job, an updated blueprint fixes the bug, for all future analysis runs of all analysis users. Importantly, updates can be made to a public analysis without influencing the reproducibility of past results (see method details and Figure S6 for details, and additional resources for links to a full developer guide).

Testing the NeuroCAAS usage model

Next, we study how the design of NeuroCAAS translates into quantifiable analysis benefits. We confirmed the accessibility of data analyses on NeuroCAAS by opening the platform to a group of alpha testers (users and developers) over a period of 22 months. In Figure 3A, we see that although some users analyzed a handful of datasets, others analyzed hundreds and spent days of compute with the platform. Figure 3B further studies the co-occurrence of different usage patterns: a large number of single dataset jobs are suggestive of one-off exploratory use, whereas there is also a considerable proportion of jobs that leverage NeuroCAAS′s capacity for parallelism, analyzing anywhere from 2 to 70 datasets in a single job. We also grouped usage by data analysis (Figure 3C). We classified different analyses as follows: dark blue bars indicate existing analysis adapted for NeuroCAAS by manuscript authors. These analyses were developed collaboratively, and in many cases, we iterated on an initial ‘‘dev’’ version of an analysis adapted for NeuroCAAS with feedback from users, before releasing a ‘‘public’’ version that was robust to various differences in workflow and dataset type. Light green bars indicate analyses developed by independent researchers following our developer workflow. We highlight the fact that analyses built following the developer workflow are well used, indicating the viability of the workflow that we have built. Dark green bars indicate analyses that we introduce in this paper specifically for NeuroCAAS infrastructure, described further in sections NeuroCAAS simplifies large data pipelines: widefield imaging protocol and NeuroCAAS stabilizes deep learning models: ensemble markerless tracking. Finally, light blue bars indicate ‘‘custom’’ analyses that we built for particular user groups. NeuroCAAS authors built custom analyses through simple copying and editing of existing, general purpose blueprints. Although per-user custom analyses are not a focus of our platform, these results demonstrate the ease with which different variants of an analysis can be provided

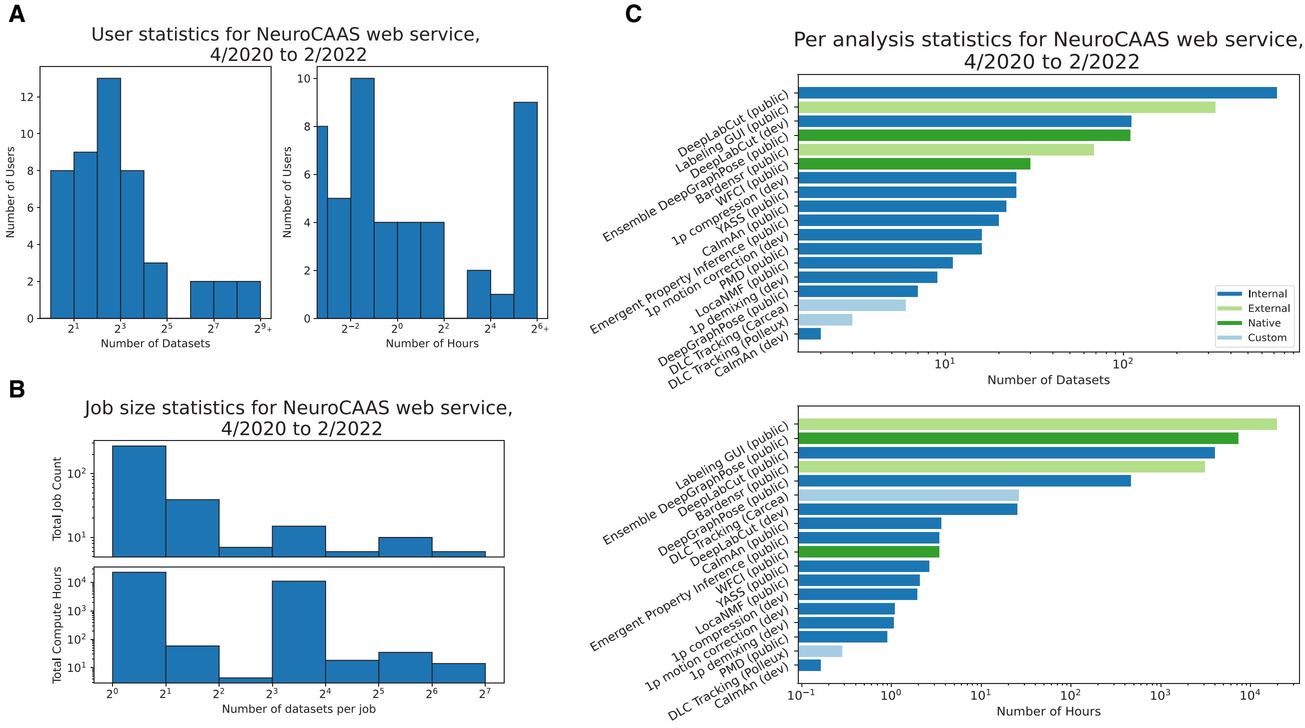


Figure 3. Usage statistics for NeuroCAAS platform over a 22-month test period

(A) Histograms for number of datasets (left) and corresponding compute hours (right) spent by each active user of NeuroCAAS.

(B) Histograms for job size indicate the number of datasets (top) and corresponding compute hours (bottom) concurrently analyzed in jobs.

(C) Usage grouped by platform developer. Dark blue (internal): analyses adapted for NeuroCAAS by paper authors. Light green (external): analyses that were not developed by NeuroCAAS authors. Dark green (native): NeuroCAAS native analyses (see NeuroCAAS simplifies large data pipelines: widefield imaging protocol and NeuroCAAS stabilizes deep learning models: ensemble markerless tracking). Light blue (custom): custom versions of generic analyses built for individual alpha users. We exclude usage attributed to analysis testing by NeuroCAAS team members.

within NeuroCAAS's design, and we discuss how users can leverage NeuroCAAS for custom use cases in the discussion.

Next, we confirmed that the design of data analysis using NeuroCAAS's IaC approach does indeed provide robust reproducibility. We selected two analyses available on our platform, and we analyzed the similarity of analysis outputs across multiple runs in Table 1 (see Figure S3 for in depth analysis, and Figure S4 for corresponding analysis of timings). Fixing a single dataset, configuration file, and blueprint for each analysis, we evaluated reproducibility of results in terms of differences between analysis outputs (see Table 1 caption for metrics). First, we observe that over multiple runs conducted by the same researcher in the United States, (Table 1, versus Run 2), results showed no scientifically relevant differences. Second, we varied the identity and physical location of the person requesting NeuroCAAS jobs: compared with jobs started by independent researchers in India and Switzerland (Table 1, versus Run 5, Run 10, respectively), we once again note no meaningful differences in the outputs of these analyses. Although physical location might bias or restrict researchers to use specific analysis infrastructure on other platforms, they have access to the exact same analysis infrastructure through NeuroCAAS. Finally, we conducted a test to measure if our platform was truly IaC: given dataset, configuration file, and analysis blueprint, there should be no reliance on the compute resources that we used to develop these analysis and perform reference runs. For a final run, we automatically deployed a complete clone of the NeuroCAAS platform on a new set of cloud resources, as any user of our platform can do in a few simple steps (details in discussion; Figure S5). We then ran jobs with the blueprints, datasets, and configuration files for the corresponding analyses (Table 1, versus Run 14), showing that results from this cloned platform are indistinguishable from those generated by the original platform.

Existing platforms leave infrastructure gaps

Although we do not attempt an exhaustive review of existing analysis platforms in neuroscience here, we characterize some exemplars in order to contrast NeuroCAAS from typical alternatives. In Figure 4, we plot a variety of popular neuroscience analyses onto a space defined by (1) their place in the adoption lifecycle and (2) corresponding infrastructure needs. We overlay several exemplar platforms on this graph, with shading representing the kinds of analyses they are able to support. The degree to which a platform's support extends to the right defines its *accessibility* or the ease with which developers and users can configure analyses on the platform and begin to process data. Accessibility is a key feature for analyses that are still early in the adoption lifecycle with active development and a growing user base. Likewise, the degree to which a platform's support extends upwards defines its *scale*, a one-dimensional

Table 1. Quantifying reproducibility via output comparisons for two analyses on NeuroCAAS

	Reference run output (comparison metric)	versus Run 2 (US)	versus Run 5 (India)	versus Run 10 (Switzerland)	versus Run 14 (platform clone)
Analysis					
CaImAn (Giovannucci et al., 2019)	spatial components (Jaccard distance)	0.0	0.0	0.0	0.0
	temporal components (RMSE)	0.0	0.0	0.0	0.0
Ensemble DeepGraphPose (NeuroCAAS stabilizes deep learning models: ensemble markerless tracking)	body part traces (RMSE)	1.2e-8	1.2e-8	2.3e-8	1.4e-8

For CaImAn (an algorithm to analyze calcium imaging data), we independently characterized differences in the spatial and temporal components recovered by the model. Differences in spatial components are measured by the average Jaccard distance over pairs of spatial components. A Jaccard distance of 0 corresponds to two spatial components that perfectly overlap. Differences in temporal components were calculated as the average root mean squared error (RMSE) taken over paired time series of component activity. For Ensemble DeepGraphPose (an algorithm to track body parts of animals during behavior from video), we considered multiple sets of outputs from a single, pretrained model. Here, RMSE takes units of pixels; hence, differences of order 1e−8 are not relevant for behavioral quantification. For both analyses, we fixed a single dataset, configuration file, and blueprint across runs. See Figures S3 and S4 for more.

approximation of the infrastructure needs for which it can provide. Although the exact positioning of these analyses and platforms is subjective and dynamic, there are general features of the analysis platform landscape that we discuss in what follows.

Local platforms like CellProfiler (Carpenter et al., 2006), Ilastik (Sommer et al., 2011) (cell-based image processing), Icy (Chaumont et al., 2012), ImageJ (Schneider et al., 2012) (generic bioimage analysis), BIDS Apps (Gorgolewski et al., 2017) (MRI analyses for brain imaging data structure format), and Bioconductor (Amezquita et al., 2020) (genomics) have all achieved success in the field by packaging together popular core analyses with necessary software dependencies and intuitive, streamlined user interfaces. Most of these local platforms also have an open contribution system for interested developers. Local platforms are thus highly accessible to both developers and users but are in the large majority of cases designed only for use on a user's local hardware, limiting their scale (Figure 4, bottom).

In contrast, remote platforms like the Neuroscience Gateway (NSG) (Sanielevici et al., 2018) (specializing in neural simulators), Flywheel (flywheel.io) (emphasizing fMRI and medical imaging), and neuroscience-focused research computing clusters offer powerful hardware through the extreme science and engineering discovery environment (XSEDE) portal (Towns et al., 2014), the public cloud and on-premises hardware, respectively. These remote platforms offer powerful compute, but at the cost of accessibility to users, who must adapt their software and workflow to new conventions (i.e., wait times for jobs to run on shared resources, hardware specific installation, custom scripting environments, and limitations on concurrency) in order to make use of offered hardware. As a particular example, NSG requires users to submit a script that they would like to have run on existing compute nodes in the XSEDE cluster, making it more similar to a traditional on-premises cluster in usage than NeuroCAAS. NSG also restricts jobs to run serially and does not have an open system for contributing new analyses, making it incompatible with the usage model and analyses that we present here. Likewise, although Flywheel (Flywheel Exchange, 2019) (with a focus on human brain imaging tools) offers the option of cloud compute, the platform is not structured in terms of infrastructure stacks for given analyses, in effect leaving many infrastructure design choices to individual users. Altogether, remote platforms are best for committed, experienced users who are already familiar with the analyses available on the remote platform and understand how to optimize them for available hardware. It is also more difficult to contribute new analyses to these platforms than their locally hosted counterparts (see Figure 4, left side). This difficulty makes them less suitable for actively developing or novel analyses, as updates may be slow to be incorporated, or introduce breaking changes to user-written scripts, making remote platforms altogether less accessible than local ones.

Some platforms provide both local or remote style usage: Galaxy (Goecks et al., 2010) and Brainlife (Avesani et al., 2019) offer a set of default compute resources but can also be used to run analyses on personal computing resources or the cloud. These mixed-compute models provide a useful way to increase the accessibility of many analyses and can provide levels of reproducibility similar to that provided by the IAE and job manager of NeuroCAAS. However, without having an IaC framework that makes a reproducible configuration of compute resources available to all analysis users, we lose the guarantee that all platform users will be able to use analyses that depend on specific infrastructure configurations. As noted in Goecks et al. (2010), it is a challenge even for these mixed-compute platforms to ensure accessibility for an analysis developed on a given set of local compute resources—significant work must be done to make this analysis functional on other computing resources or to maintain these local compute resources in order to ensure that others can use them whenever needed. These challenges are only exacerbated by the increasing reliance of analysis tools upon more powerful and specific infrastructure configurations, such as high performance GPUs.

Although undeniably useful, all available platforms operate on a tradeoff that forces researchers to choose between accessibility and scale. Although these platforms often concentrate on applications that mitigate the effects of this tradeoff, there are

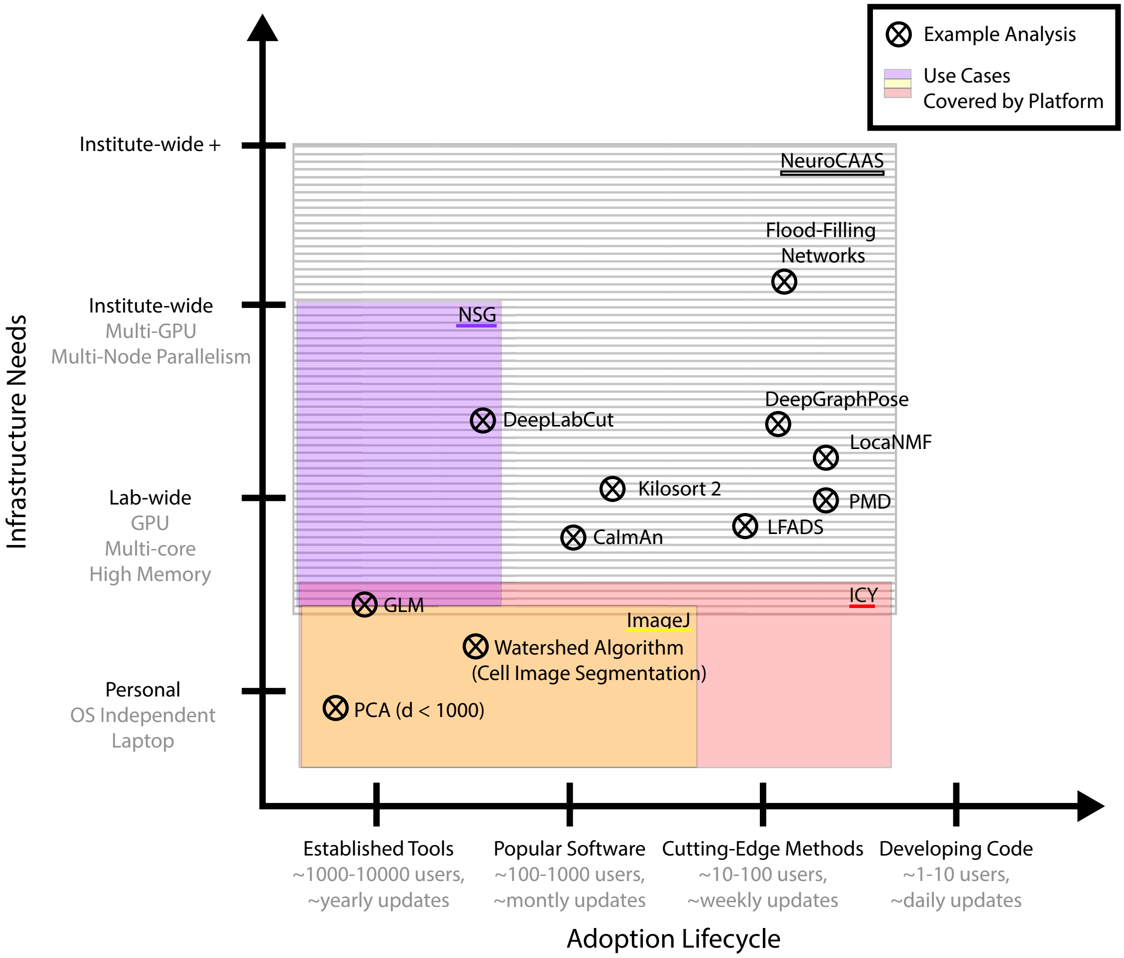


Figure 4. Landscape of cellular/systems-level neuroscience analysis platforms

Crosses: popular analyses in terms of their place in the adoption lifecycle (number of users, rate of software updates) and their infrastructure needs. Coloring: representative platforms, indicating the parts of analysis space that are covered by a given platform. (Example analyses: [Goodman and Brette, 2009; Pnevmatikakis et al., 2016; Mathis et al., 2018; Pachitariu et al., 2016, Pandarinath et al., 2018; Januszewski et al., 2018; Saxena et al., 2020; Buchanan et al., 2018; Graving et al., 2019]; representative platforms: [Sanielevici et al., 2018; Chaumont et al., 2012; Schneider et al., 2012]).

many popular analyses that would not be suitable for existing analysis platforms (see Figure 4, center). Furthermore, critically for reproducibility, across all existing platforms, analysis users and developers are still required to manually configure analysis infrastructure, whether by installing new tools onto one's personal infrastructure or porting code and dependencies to run in a remote (and sometimes variably allocated) infrastructure stack.

Some platforms in cellular and molecular biology (Riley, 2010), as well as bioinformatics (Simonyan and Mazumder, 2014; Terra, 2022) have shown that IaC approaches are feasible to handle these infrastructure issues. A notable difference in the design of our platform is that these other platforms assume that individual users will themselves manually configure analyses—that they will choose relevant resources to support the analyses that they want to conduct or compose an analysis of interest out of a set of small modular parts; in each case performing the work of a NeuroCAAS developer. Although resulting analyses can be shared with other individuals on a case-by-case basis, this is very different from NeuroCAAS. Our platform is geared toward a heterogeneous community of researchers, where some researchers are developing general purpose analyses that only need to be configured or updated once before being used by a large group of potential users.

Next, we concretely demonstrate how the infrastructure benefits of the NeuroCAAS platform address ongoing challenges in neuroscience data analysis. We show two examples of NeuroCAAS native analyses that would not be feasible without the simultaneous benefits to accessibility, scale, and reproducibility that we provide.

NeuroCAAS simplifies large data pipelines: Widefield imaging protocol

Often, big data pipelines demand many individual preprocessing steps, creating the need for unwieldy multi-analysis infrastructure stacks—infrastructure stacks that support the needs of multiple core analyses at the same time. A notable example is

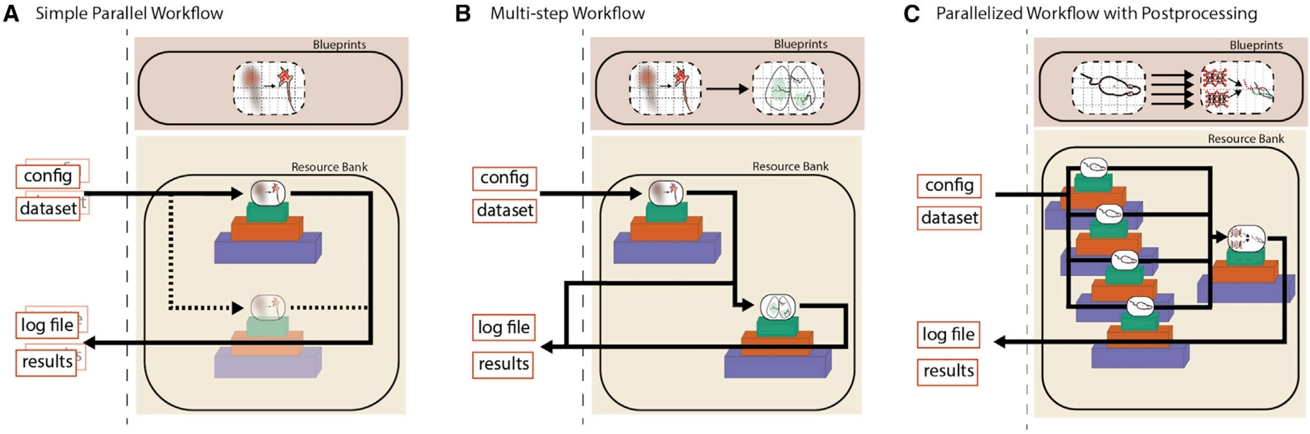


Figure 5. NeuroCAAS supports multi-stack design patterns

(A) Default workflow: if more than one dataset is submitted, NeuroCAAS automatically creates separate infrastructure for each.

(B) Chained workflow: multiple analysis components with different infrastructure needs are seamlessly combined on demand. Intermediate results are returned to the user so that they can be examined and visualized as well (NeuroCAAS simplifies large data pipelines: widefield imaging protocol).

(C) Parallelism + chained workflow: workflows A and B can also be combined to support batch processing pipelines with a separate postprocessing step (NeuroCAAS stabilizes deep learning models: ensemble markerless tracking).

widefield calcium imaging (WFCI)—a high-throughput imaging technique that can collect activity-dependent fluorescence signals across the entire dorsal cortex of an awake, behaving mouse (Couto et al., 2021), potentially generating terabytes of data across chronic experiments. The protocol paper Couto et al. (2021) describes a complete WFCI analysis that links together cutting-edge data compression/denoising with demixing techniques designed explicitly for WFCI (via penalized matrix decomposition, or PMD [Buchanan et al., 2018] and localized semi-nonnegative matrix factorization [LocaNMF] [Saxena et al., 2020], respectively). Each of these analyses depends upon its own specialized hardware and installation, creating many competing requirements on a multi-analysis infrastructure stack that are difficult to satisfy in practice. Although we offer a NeuroCAAS implementation of the described WFCI analysis in Couto et al. (2021), we do not discuss how NeuroCAAS addresses the issue of multi-analysis infrastructure stacks, which can pose IaGS challenges even to our blueprint based infrastructure.

Instead of working with multi-analysis infrastructure stacks, for this analysis NeuroCAAS extends the function of a standard job manager (see Figure 5A) to trigger multiple jobs, built from separate blueprints, in sequence (Figure 5B). We employ this design to dramatically simplify the infrastructure requirements for a complete WFCI pipeline. First the initial steps of motion correction, denoising, and hemodynamic correction of the data are run from a blueprint that emphasizes multicore parallelism (64 CPU cores) to suit the matrix decomposition algorithms employed by PMD. Upon termination of this first step, analysis results are not only returned to user storage, but also used as inputs to a second job, performing demixing with LocaNMF on infrastructure supporting a high performance GPU. This modular organization improves the performance and efficiency of each analysis component (see Figure 7), and also allows users to run steps individually if desired, giving them the freedom to interleave existing analysis pipelines with the components offered here. As an alternative to the standard NeuroCAAS interface, this WFCI analysis can be controlled from a custom-built graphical user interface (GUI). This GUI further extends NeuroCAAS′s accessibility with features such as interactive alignment of a brain atlas to user data as part of parameter configuration (in the process validating input data as well). Following parameter configuration, this GUI interacts with NeuroCAAS *programmatically*, using locally run code scripts to perform data upload and job submission, and to detect and retrieve results once analysis is complete. Results can be visualized directly in this GUI as well. Altogether, this GUI can be used as a model for researchers who would like to take advantage of our computational infrastructure within a more sophisticated user interface, or integrate NeuroCAAS programmatically with other software tools. Importantly, despite its interactivity, the performance of our WFCI analysis does not depend on the infrastructure available to the user. For example, users could simultaneously launch many analyses and have them run in parallel through this GUI, easily conducting a hyperparameter search over their entire multi-step analysis. Researchers can find the GUI for this WFCI analysis with NeuroCAAS integration at https://github.com/jcouto/wfield.

For developers, this analysis presents a counterpart to existing domain specific projects such as CaImAn (Giovannucci et al., 2019) for cellular resolution calcium imaging or SpikeInterface (Buccino et al., 2020) for electrophysiology, which explicitly make multi-step data analyses compatible with optimized hardware. In contrast, our WFCI analysis is made directly available to users on powerful remote hardware without the need for anyone to revise existing analysis or infrastructure. To our knowledge, there is no other neuroscience platform with the accessibility, scale, and support for reproducibility to link together cutting-edge analyses across separate infrastructures and make this exact configuration available directly to the research community.

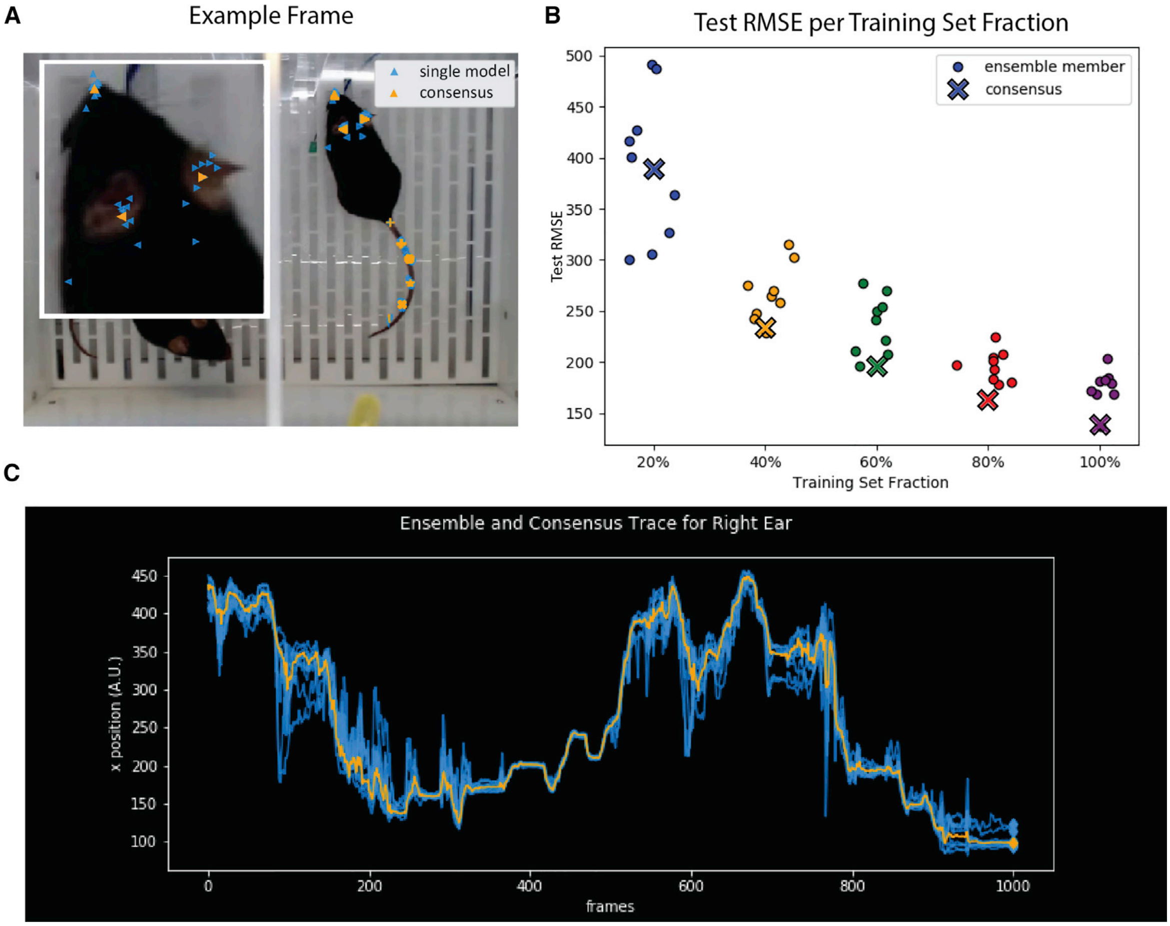


Figure 6. Ensemble markerless tracking

(A) Example frame from mouse behavior dataset (courtesy of Erica Rodriguez and C. Daniel Salzman) tracking keypoints on the top down view of a mouse, as analyzed in Wu et al. (2020). Marker shapes track different body parts: blue markers representing the output of individual tracking models, and orange markers representing the consensus. Inset image shows tracking performance on the nose and ears of the mouse.

(B) Consensus test performance versus test performance of individual networks on a dataset with ground truth labels as measured via root mean squared error (RMSE).

(C) Traces from 9 networks (blue) + consensus (orange). Across the entire figure, ensemble size = 9. (A) and (C) correspond to traces taken from the 100% split in (B) corresponding to 20 training frames.

NeuroCAAS stabilizes deep learning models: Ensemble markerless tracking

The black box nature of deep learning can generate sparse, difficult to detect errors that reduce the benefits of deep learning based tools in sensitive applications. For modern markerless tracking analyses built on deep neural networks (Mathis et al., 2018; Graving et al., 2019; Nilsson et al., 2020; Wu et al., 2020), these errors can manifest as "glitches" (Wu et al., 2020), where a marker point will jump to an incorrect location, often without registering as an error in the network's generated likelihood metrics (see Figure 6).

One general purpose approach to combat the unreliable nature of individual machine learning models is ensembling (Dietterich, 2000): instead of working with a single model, a researcher simultaneously prepares multiple models on the same task, subsequently aggregating their outputs into a single consensus output. Ensemble methods have been shown to be effective for deep networks in a variety of contexts (Lakshminarayanan et al., 2017; Fort et al., 2019; Ovadia et al., 2019), but they confer a massive infrastructure burden if run on limited local compute resources: researchers must simultaneously train, manage, and aggregate outputs across many different deep learning models, incurring either prohibitively large commitments to deep learning specific infrastructure and/or infeasibly long wait times.

In contrast, NeuroCAAS enables easy and routine implementation of ensemble methods. By modifying the NeuroCAAS job manager, we designed an analysis that takes input training data and distributes it to N identical sets of IAEs and resource bank instances (Figure 5C). For the application shown here, we used an IAE with DeepGraphPose (Wu et al., 2020) as our core analysis; the N infrastructures differ only in the minibatch order of data used to train models. The results from each trained model are then used to produce a consensus tracking output, taking each individual model's estimate of part location across the entire image (i.e., the confidence map output) and averaging these estimates. Even with this relatively simple approach, we find the consensus tracking output is robust to the errors made by individual models (Figures 6A and 6C). This consensus performance is maintained even when we significantly reduce the size of the training set (Figure 6B). Finally, in Figure 6C, we can see that there are portions of the dataset where the individual model detections fluctuate around the consensus detection. This fluctuation offers an empirical readout of tracking difficulty within any given dataset; frames with large diversity in the ensemble outputs are good candidates for further labeling and could be easily incorporated in an active learning loop. After training, models can be kept in user storage and used to analyze further behavioral data, without moving these models out of NeuroCAAS. Overall, Figure 6 shows that with the scale of infrastructure available on NeuroCAAS, ensembling can easily improve the robustness of markerless tracking, naturally complementing the infrastructure reproducibility provided by the platform.

NeuroCAAS is uniquely capable of providing the flexible infrastructure necessary to support a generally available, on-demand ensemble markerless tracking application. To our knowledge, none of the platforms with the scale to support markerless tracking on publicly available resources (e.g., on premise clusters, Google Colab, Galaxy [Goecks et al., 2010], NSG [Sanielevici et al., 2018], Brainlife [Avesani et al., 2019]) can satisfactorily alleviate the burden of a deep ensembling approach, still forcing the user to accept either long wait times or manual management of infrastructure. These limitations also prohibit use cases involving the quantification of ensemble behavior across different parameter settings (cf. Figure 6B, where we trained 45 networks simultaneously).

NeuroCAAS is faster and cheaper than IaGS analogs

NeuroCAAS offers a number of major advantages over IaGS: reproducibility, accessibility, and scale, whether we compare against a personal workstation or resources allocated from a locally available cluster. However, since NeuroCAAS is based on a cloud computing architecture, one might worry that data transfer times (i.e., uploading and downloading data to and from the cloud) could potentially lead to slower overall processing or that the cost of cloud compute could outweigh that of local infrastructure.

Figure 7 considers this question quantitatively, comparing NeuroCAAS with a simulated personal workstation (see quantification and statistical analysis for details). For the analogous comparisons (with similar conclusions) against a simulated local cluster, see Figure S7. Figure 7 presents time and cost benchmark results on four popular analyses that cover a variety of data modalities: CaImAn (Giovannucci et al., 2019) for cellular resolution calcium imaging, DeepLabCut (DLC) (Mathis et al., 2018) for markerless tracking in behavioral videos, and a two-step analysis consisting of PMD (Buchanan et al., 2018) and LocaNMF (Saxena et al., 2020) for analysis of widefield imaging data. To be (extremely) conservative, we assume local infrastructure is set up, neglecting all of the time associated with installing and maintaining software and hardware.

Across all analyses and datasets considered in Figure 7, analyses run on NeuroCAAS were significantly faster than those run on the selected local infrastructure, even accounting for the time taken to stage data to the cloud (Figure 7A, left panes). We batched data to take advantage of both compute optimization offered by individual core analyses and NeuroCAAS ′s scale (see quantification and statistical analysis for details). These examples show that many analyses can be used efficiently on NeuroCAAS, regardless of the degree to which they have been intrinsically optimized for parallelism. Additionally NeuroCAAS upload time can be ignored if analyzing data that is already in a user storage—for example if there is a need to reprocess data with an updated algorithm or parameter setting—leading to further speedups. Finally, although we found download speeds negligible (see quantification and statistical analysis for full details of timing quantifications), this could vary significantly based on user internet speeds and analyses. Across our platform, we have attempted to design analyses with much smaller outputs than input data—a point we will return to in the discussion.

Next, we turn to cost analyses. Over the range of algorithms and datasets considered here, we found that the total baseline NeuroCAAS analysis cost was on the order of a few US dollars per dataset (Figure 7A, right panels)—see Table S7 for pricing details. We observe that for the most part, costs are approximately linear in compute time, ensuring that even compute intensive operations like training a deep network for DLC (~12 h on the instances used here) can be accomplished for ~$10—trained networks can also be maintained in cloud storage to reduce data transfer cost. In addition to our baseline implementation, we also offer an option to run analyses at a significantly lower price (indicated as "Std" and "Save," respectively, in the cost barplots in Figure 7), if the user can upper bound the expected runtime of their analysis to anything lower than 6 h (i.e., from previous runs of similar data).

Finally, we compare the cost of NeuroCAAS directly with the cost of purchasing local infrastructure. We use a total cost of ownership (TCO) metric (Morey and Nambiar, 2009) that includes the purchase cost of local hardware, plus reasonable maintenance costs over estimates of hardware lifetime; see quantification and statistical analysis for full details. We first ask how frequently one would have to run the analyses presented in Figure 7 before it becomes worthwhile to purchase dedicated local infrastructure. This question is answered by the local cost crossover (LCC): the threshold weekly rate at which a user would have to analyze data for NeuroCAAS costs to exceed the TCO of local hardware. As an example, the top two bars of Figure 7B, left, show that in order for a local machine to be cost effective for CaImAn, one must analyze ~100 datasets of 8.39 GB per week, every week for several years (see Table S4 for a conversion to data dimensions). In all use cases, the LCC rates in Figure 7B show that a researcher would have to consistently analyze ~10–100 datasets per week for several years before it becomes cost effective to use local infrastructure. Although such use cases are certainly feasible, managing these use cases on local infrastructure via IaGS would involve an significant amount of human labor.

In Figure 7C, we characterize this labor cost via the local utilization crossover (LUC): the actual time cost of analyzing data on

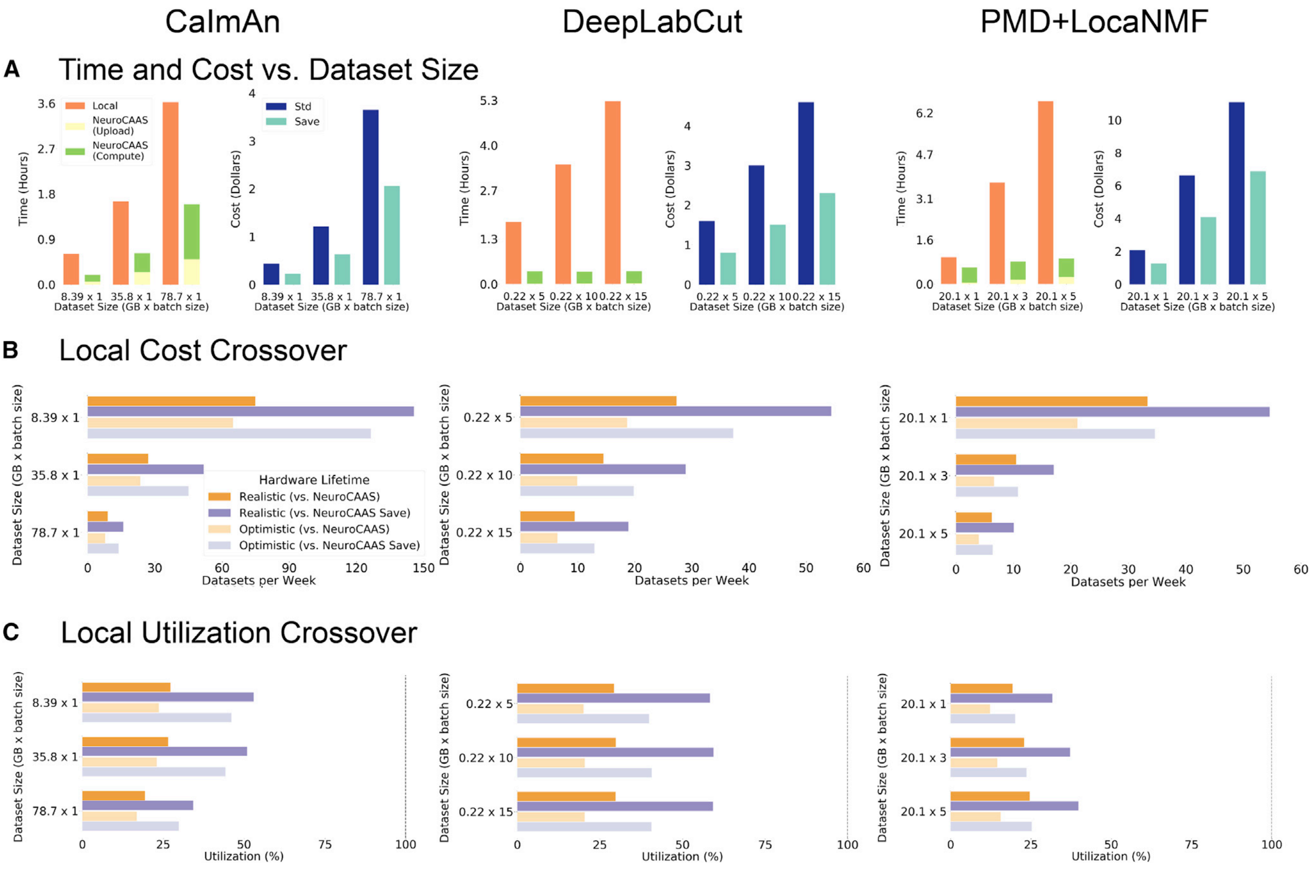


Figure 7. Quantitative comparison of NeuroCAAS versus local processing for three different analyses

(A) Simple quantifications of NeuroCAAS performance. Left graphs compare total processing time on NeuroCAAS versus local infrastructure (orange). NeuroCAAS processing time is broken into two parts: upload (yellow) and compute (green). Right graphs quantify cost of analyzing data on NeuroCAAS with two different pricing schemes: standard (dark blue) or save (light blue).

(B) Cost comparison with local infrastructure (LCC). Figure compares local pricing against both standard and save prices, with realistic (2 years) and optimistic (4 years) lifecycle times for local hardware.

(C) Achieving crossover analysis rates. Local utilization crossover gives the minimum utilization required to achieve crossover rates shown in (B). Dashed-vertical line indicates maximum feasible utilization rate at 100% (utilizing local infrastructure 24 h every day). See Figure S7 for cluster analysis, and Tables S4–S8 for supplemental information.

a local machine at the corresponding LCC rate. Across the analyses that we considered, local infrastructure would have to be dedicated to the indicated analysis for 25%–50% of the infrastructure's total lifetime (i.e., ~6–12 h per day, every day) to achieve its corresponding LCC threshold, requiring an inordinate amount of work on the part of the researcher to manually run datasets, monitor analysis progress for errors, or build the computing infrastructure required to automate this process—in essence forcing researchers to perform by hand the large-scale infrastructure management that NeuroCAAS achieves automatically. These calculations demonstrate that even without considering all of the IaGS issues that our solution avoids or explicitly assigning a cost to researcher time, it is difficult to use local infrastructure more efficiently than NeuroCAAS for a variety of popular analyses. Given the diversity of IaGS solutions, we also provide a tool for users to benchmark their available infrastructure options against NeuroCAAS (see the instructions at https://github.com/cunningham-lab/neurocaas).

NeuroCAAS is offered as a free service for many users

In many cases, researchers may use infrastructure available on hand to test out analyses before purchasing components for a dedicated infrastructure stack. Given the low per-dataset cost and the major advantages summarized above of NeuroCAAS compared with the current IaGS status quo, we have decided to mirror this model on the NeuroCAAS platform, and subsidize a large part of NeuroCAAS usage by the community. Users do not need to setup any billing information or worry about incurring any costs when starting work on NeuroCAAS; we cover all costs up to a per-user cap (initially set at $300). This subsidization removes one final friction point that might slow adoption of NeuroCAAS, and protects NeuroCAAS as a non-commercial open-source effort. Since NeuroCAAS is relatively inexpensive, many users will not hit the cap; thus, for these users, NeuroCAAS is offered as a free service. We note that we are also open to considering budget increases for researchers as they become necessary.

DISCUSSION

NeuroCAAS integrates rigorous infrastructure practices into neural data analysis while also respecting current development and use practices. The fundamental choice made by NeuroCAAS is to provide analysis infrastructure with as much automation as possible. This choice naturally makes NeuroCAAS into a *service*, and in the simplest case, neither analysis users nor analysis developers have to manage infrastructure directly; rather, NeuroCAAS removes the infrastructure burden entirely. However, as an open-source project, NeuroCAAS also acknowledges the possibility that some users may want to accept some degree of responsibility for computing infrastructure, in return for a greater degree of flexibility in how they use the platform. We highlight two notable alternative use cases here:

Working at scale: Large datasets/many jobs

Although our drag-and-drop console removes the need for users to have previous experience with coding, some users may find the console restrictive when working with large datasets or managing many jobs at once—both important facets of analysis use that NeuroCAAS is poised to improve. These restrictions can be reduced by working with NeuroCAAS using a command line interface (CLI), or by integrating calls to NeuroCAAS within locally run applications, as is done in section NeuroCAAS simplifies large data pipelines: widefield imaging protocol. Since NeuroCAAS can be used solely by interacting with cloud storage, these interfaces to NeuroCAAS are easily supported by general purpose data transfer tools. We provide instructions for this use case in our developer documentation (see additional resources).

In order to streamline data transfer in cases where input or output data are unavoidably large, we have also implemented a ‘‘storage bypass’’ option for our CLI interface. Using this option, public data stored elsewhere in the AWS cloud can be analyzed, and results can be written back directly to this location without incurring additional data transfer time and costs, laying the groundwork for the integration of NeuroCAAS analyses with external data sources. This option is intended for analyses that handle especially large input or output data, where CLI use is preferable, but we plan to extend this functionality to all analyses and our standard interface soon. We believe these additional features will better equip NeuroCAAS to handle the ever increasing scale of neuroscience data (e.g., Steinmetz et al., 2021; Couto et al., 2021), as well as methods that consider multiple data modalities simultaneously (e.g., Batty et al., 2019), and facilitate sharing of analysis outputs across many users.

Working independently: private management of costs/compute resources

A major benefit of NeuroCAAS′s IaC construction is that the entire platform (except private user data) can be reconstructed automatically, given the code in the NeuroCAAS source repository (method details; Figure S5). There is no dependence of the platform upon specifics of infrastructure configuration that are not recorded in a blueprint. This benefit means that if users anticipate very high costs or would like to use IaC to manage their own custom analyses, it is easy for them to switch from using our public implementation of NeuroCAAS to one that they pay for themselves, maintaining all the benefits of NeuroCAAS′s infrastructure management. We provide detailed instructions on this process in our developer documentation (additional resources), describing platform setup and cloning of individual analyses.

Finally, we revisit NeuroCAAS′s stated objectives of supporting reproducible, accessible, and scalable data analyses. These are fundamentally multifaceted issues and will manifest in different ways across a variety of use cases. To this end, we identify strengths and limitations of NeuroCAAS′s approach to these issues (and related costs) so that researchers can evaluate the suitability of NeuroCAAS to their particular use case.

Reproducibility

What are the benefits and limits of analysis reproducibility in NeuroCAAS? In the section testing the NeuroCAAS usage model, we show that a dataset, configuration file, and analysis blueprint form a set of sufficient resources to reproduce an analysis output against a set of practically relevant interventions. We note some qualifications to this performance: First, it can be nontrivial to maintain a dataset across multiple analysis runs. Importantly, when data are uploaded, it will not be versioned by default, creating the potential for it to be overwritten. For dataset provenance, we recommend data infrastructure projects like DANDI (Dandi Team, 2019). Integration with a data archive is an important future direction to extend reproducibility for NeuroCAAS. Second, we are limited by the inherent computational reproducibility of the core analysis we offer—for example, random computations can introduce significant differences from run to run (although ensemble methods can mitigate these issues). Finally, we can consider the life cycle of different resources on the AWS cloud. For example, reproducibility could be affected if support for certain hardware instances become deprecated, and can no longer be used to run analyses. Given the large scale reliance of industrial applications on the AWS cloud, such events are very rare and announced well in advance, but we can take steps to address such a contingency. In particular, an important future direction is to consider how we can expand our approach outside of a particular cloud provider (see method details for more).

Accessibility

NeuroCAAS aims to improve the accessibility of data analysis by removing the need for users to independently configure infrastructure stacks, as is the *de facto* standard with IaGS approaches. By default, NeuroCAAS does not aim to improve other aspects of accessibility, such as the scientific use of core analysis algorithms. For example, if a user has data that are incorrectly formatted for a particular algorithm, the same error will happen with NeuroCAAS as it would with conventional usage, even though curated deployments and blueprint based updates can significantly mitigate such issues.

Another approach toward achieving robust and general purpose analyses focuses on the explicit standardization of data formats and workflow. As mentioned, we plan to integrate with data archiving projects like distributed archives for neurophysiology data integration (DANDI) (Dandi Team, 2019) that enforces the

Neurodata Without Borders (NWB) (Teeters et al., 2015; Rübel et al., 2019, 2021) data standard, providing a stable set of expectations for analysis developers while also improving reproducibility of analysis results. Likewise, workflow management systems for neuroscience such as Datajoint (Yatsenko et al., 2015) or more general tools like snakemake (Köster and Rahmann, 2012) and the Common Workflow Language (Amstutz et al., 2016) codify the sequential steps that make up a data analysis workflow on given infrastructure, ensuring data integrity and provenance. Other platforms both within (NeuroScout, 2022; Avesani et al., 2019) and outside of neuroscience (Seven Bridges Genomics, 2019) provide well-designed examples of how standardized data formats paired with workflow management systems can be used to make analyses more modular and easy to use.

Scale

Although NeuroCAAS offers analyses at scale, it does not offer unstructured access to cloud computational resources. The concept of IAEs should clarify this fact: NeuroCAAS serves a set of analyses that are configured to a particular specification, as established by the analysis developer. This constraint is often ideal, since the specification is in many cases established by the analysis method's original authors. Without specific structure to manage the near infinite scale of resources available on the cloud, the management of resources on the cloud easily becomes susceptible to the issues of IaGS that motivated the development of NeuroCAAS to begin with (Monajemi et al., 2019). The constraint of immutability distinguishes NeuroCAAS from interactive data analysis offerings that offer cloud computing like Pan-neuro (Rokem et al., 2021) or Google Colab, in keeping with their differing intended use cases. Although interactive computing plays a key role in data analysis applications, we believe there is fundamental value in immutable data analyses as well.

Importantly, immmutability does not suggest that analyses on NeuroCAAS are a black box. All NeuroCAAS analyses are built from open-source projects, the workflow scripts used to parse datasets and config files inside an IAE are made available to all analysis users, and jobs constantly print live status logs back to users. Furthermore, our novel analyses show that there are means of comprehensively characterizing analysis performance that only become available at scale (i.e., full parameter searches over a multi-step analysis or ensembling to evaluate reliability of analysis outputs).

Cost

The cost quantifications that we present in this manuscript are intended to demonstrate that the cost of using NeuroCAAS's computing infrastructure is practically feasible when compared against the cost of computing on typical IaGS infrastructure. One point to note is that for individual research groups, the cost of using local infrastructure may vary significantly across institutions. Our quantifications are best fit to the case where a research group is supporting its own computing costs and resources. Although the relative cost of using NeuroCAAS may thus differ from group to group, it is our hope that offering analyses at a uniform (and highly subsidized) cost will increase analysis accessibility to a significant portion of the neuroscience community and potentially provide a more concrete understanding of the costs associated with the development and adoption of new analysis tools.

Beyond compute, we do not discuss the costs of storing and retrieving data from the cloud in depth. Without restricting data sizes on user storage, we found that data storage costs were small enough that we could support them without counting them toward user budgets. A common theme of the analyses that we discuss is that we can minimize data retrieval costs by designing workflows such that analysis results that the user actually needed to retrieve were far smaller than input data, specifically by modifying IAEs and by maintaining large intermediate results on the cloud for use in future analyses. NeuroCAAS's cost benefits may be reduced if these conditions are harder to achieve for a given analysis, although we believe that alternative use cases, such as our CLI interface with "storage bypass" are well poised to handle these contingencies, especially when paired with future directions such as integration with a data archive. Importantly, on private resources users will have to pay for cloud storage. This cost can be minimized by deleting input data and storing all results locally when not in use.

Beyond our proposed improvements above, NeuroCAAS will naturally continue to evolve by virtue of its open-source code and public cloud construction. First, we hope to build a community of developers who will add more analysis algorithms to NeuroCAAS, with an emphasis on subfields of computational analysis that we do not yet support. Throughout this manuscript, we focus largely on analyses in systems neuroscience and neurophysiology, in accordance with the previous experience of the authors, and the opinion that analyses in this area are in great need of the platform design implemented by NeuroCAAS. We also plan to add support for real-time processing (e.g., Giovannucci et al. [2017] for calcium imaging or Schweihoff et al. [2021] and Kane et al. [2020] for closed-loop experiments, or Lopes et al. [2015] for the coordination of multiple data streams), using blueprint based methods to design fast, reliable infrastructure for closed-loop analyses, in the same spirit as these batch mode analyses. Second, other tools have brought large-scale distributed computing to neural data analyses (Freeman, 2015; Rocklin, 2015) in ways that conform to more traditional high performance computing ideas of scalability for applications that are less easily parallelized than those presented here. Integrating more elaborate scaling into NeuroCAAS while maintaining development accessibility will be an important goal going forward. Third, we aim to take inspiration from other computing platforms both within and beyond neuroscience to improve the usability of our platform, such as reporting the expected runtime and success rate of analyses (Avesani et al., 2019), indicating the compatibility of different analysis steps in a sequence (Seven Bridges Genomics, 2019) or improving user and developer resources to include forums and full-time support (Goecks et al., 2010). We also aim to identify platforms and tools that could potentially be integrated with NeuroCAAS resources, in order to provide them with the infrastructure reliability that we prioritize. Finally, a major opportunity for future work is the integration of NeuroCAAS with modern visualization tools. We have emphasized above that IAEs on

NeuroCAAS are designed with the ideal of fully automated data analyses in mind, because of the virtues that automation brings to data analyses. However, we recognize that for some of the core analyses on NeuroCAAS, and indeed most of those popular in the field, some user interaction is required to speed up analysis and optimize results. We have already demonstrated the compatibility of interactive interfaces with NeuroCAAS in our WFCI analysis, and we will aim to establish a general purpose interface toolbox for developers in the same spirit, without sacrificing the benefits of cost efficiency, scalability, and reproducibility that distinguish NeuroCAAS in its current form.

Longer term, we hope to build a sustainable and open-source user and developer community around NeuroCAAS. We welcome suggestions for improvements from users, and new analyses as well as extensions from interested developers, with the goal of creating a sustainable community-driven resource that will enable new large-scale neural data science in the decade to come.

STAR★METHODS

Detailed methods are provided in the online version of this paper and include the following:

- KEY RESOURCES TABLE
- RESOURCE AVAILABILITY
  - ○ Lead contact
  - ○ Materials availability
  - ○ Data and code availability
- METHOD DETAILS
  - ○ NeuroCAAS architecture specifics
  - ○ Developer workflow
  - ○ Novel analyses
- QUANTIFICATION AND STATISTICAL ANALYSIS
  - ○ Quantifying reproducibility on NeuroCAAS
  - ○ Quantifying usage
  - ○ Benchmarking algorithms on NeuroCAAS
  - ○ Survey of analyses and platforms
- ADDITIONAL RESOURCES

SUPPLEMENTAL INFORMATION

Supplemental information can be found online at https://doi.org/10.1016/j.neuron.2022.06.018.

ACKNOWLEDGMENTS

We thank Ioana Carcea and Robert C. Froemke for the use of benchmarking data for DLC, Simon Musall and Anne Churchland for the use of benchmarking data for PMD and LocaNMF, and Erica Rodriguez and C. Daniel Salzman for the use of data to demonstrate ensemble markerless tracking. We thank Larry Abbott for feedback and support regarding the goals and usage of our platform, Danil Tyulmankov for useful references and discussions of benchmarking, Dan Biderman and Michelle Stackmann for paper comments, and Jian Wang and Yakov Stern for helpful discussions on NeuroCAAS′s architecture and extensions. We thank Dmitri Yatsenko, Edgar Walker, Raphael Guzman, Thinh Nguyen, and other members of the Datajoint team, as well as Ben Dichter, Alessio Buccino and other members of the CatalystNeuro and DANDI teams for discussions of how our platform might integrate with others, as well as general advice. We thank Anjali Agarwal and Catalin Mitelut for submitting NeuroCAAS jobs as independent researchers to study reproducibility. We thank Peter Lee, Jackson Loper, Shuonan Chen, and Nick Greenspan for development of additional algorithms early in development and for help prototyping relevant extensions for job managers; Selmaan Chettih for suggesting ensembling of pose tracking; and Zahra Adahman, Ioana Carcea, Claire Everett, Andres Bendesky, Andres Villegas, Franck Polleux, Vivek Athalye, Darcy Peterka, and Avner Wallach for discussion and feedback on the use of NeuroCAAS during development; as well as all users and developers during alpha testing.

T.A. is supported by NIH training grant 2T32NS064929-11, NSF DBI-1707398, and the Gatsby Charitable Foundation (Gatsby Charitable Foundation GAT3708). S.S. is supported by the Swiss National Science Foundation P2SKP2_178197 and NIH 5U19NS104649. E.K.B. is supported by NIH training grant T32NS064929, NIH U19NS107613, NSF DBI-1707398, and the Gatsby Charitable Foundation (Gatsby Charitable Foundation GAT3708). J.P.C. is supported by Simons 542963. L.P. is funded by IARPA MICRONS D16PC00003, NIH 5U01NS103489, 5U19NS107613, 1UF1NS107696, 1UF1NS108213, 1RF1MH120680, 1U19NS123716-01, DARPA NESD N66001-17-C-4002, and Simons Foundation 543023. L.P. and J.P.C. are supported by NSF Neuronex Award DBI-1707398 and NIH 5U19NS104649.

AUTHOR CONTRIBUTIONS

T.A. and J.P.C. conceptualized the project. T.A. designed the infrastructure platform with input from all authors. S.S., I.K., S.L.K., R.G., J.Z., and T.A. developed analyses to work on the platform as well as general-purpose developer tools (with supervision from L.P. and J.P.C.), and S.S., I.K., and T.A. collected the data shown in Figure 4. J.C., S.S., and I.K. developed code for WFCI analysis and GUI, E.K.B. and T.A. developed code for ensembled markerless tracking with supervision from L.P., and J.B. developed and maintained the website. T.A., L.P., and J.P.C. wrote the paper with input from all authors.

DECLARATION OF INTERESTS

The authors declare no competing interests.

Received: June 28, 2021
Revised: May 6, 2022
Accepted: June 22, 2022
Published: July 22, 2022

SUPPORTING CITATIONS

The following reference appears in the Supplemental Information: Whiteway et al., 2021; Chen et al., 2020; Minka, 1999; Sussillo et al., 2016; Bittner et al., 2019

REFERENCES

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. (2016). Tensorflow: a system for large-scale machine learning. In 12th Symposium on Operating Systems Design and Implementation (OSDI 16), pp. 265–283.

Aguiar, A., Díaz, J., Almaraz, R., Pérez, J., and Garbajosa, J. (2018). DevOps in practice—an exploratory case study. In Proceedings of the 19th International Conference on Agile Software Development: Companion, XP ’18, pp. 1–3.

Amezquita, R.A., Lun, A.T.L., Becht, E., Carey, V.J., Carpp, L.N., Geistlinger, L., Marini, F., Rue-Albrecht, K., Risso, D., Soneson, C., et al. (2020). Orchestrating single-cell analysis with bioconductor. Nat. Methods *17*, 137–145. https://doi.org/10.1038/s41592-019-0654-x.

Amstutz, P., Crusoe, M.R., Tijanic, N.s., Chapman, B., Chilton, J., Heuer, M., Kartashov, A., Kern, J., Leehr, D., Menager, H., et al. (2016). Common workflow language, v1.0. https://doi.org/10.6084/m9.figshare.3115156.v2.

Avesani, P., McPherson, B., Hayashi, S., Caiafa, C.F., Henschel, R., Garyfallidis, E., Kitchell, L., Bullock, D., Patterson, A., Olivetti, E., et al. (2019). The open diffusion data derivatives, brain data upcycling via integrated publishing of derivatives and reproducible open cloud services. Sci. Data *6*, 69. https://doi.org/10.1038/s41597-019-0073-y.

Batty, E., Whiteway, M., Saxena, S., Biderman, D., Abe, T., Musall, S., Gillis, W., Markowitz, J., Churchland, A., Cunningham, J.P., et al. (2019). BehaveNet: nonlinear embedding and Bayesian neural decoding of behavioral videos. In Advances in Neural Information Processing Systems, pp. 15680–15691.

Batty, E., Merel, J., Brackbill, N., Heitman, A., Sher, A., Litke, A., Chichilnisky, E., and Paninski, L. (2016). Multilayer recurrent network models of primate retinal ganglion cell responses. In International Conference on Learning Representations.

Bittner, S.R., Palmigiano, A., Piet, A.T., Duan, C.A., Brody, C.D., Miller, K.D., and Cunningham, J.P. (2019). Interrogating theoretical models of neural computation with deep inference. Preprint at bioRxiv. https://doi.org/10.1101/837567.

Bloch, J. (2008). Effective Java (the Java Series) (Prentice Hall PTR).

Brikman, Y. (2019). Terraform: Up and Running: Writing Infrastructure as Code (O'Reilly Media).

Buccino, A.P., Hurwitz, C.L., Garcia, S., Magland, J., Siegle, J.H., Hurwitz, R., and Hennig, M.H. (2020). SpikeInterface, a unified framework for spike sorting. eLife *9*, e61834. https://doi.org/10.7554/eLife.61834.

Buchanan, E.K., Kinsella, I., Zhou, D., Zhu, R., Zhou, P., Gerhard, F., Ferrante, J., Ma, Y., Kim, S., Shaik, M., et al. (2018). Penalized matrix decomposition for denoising, compression, and improved demixing of functional imaging data. Preprint at arXiv. arXiv:1807.06203. https://doi.org/10.1101/334706.

Buckheit, J.B., and Donoho, D.L. (1995). Wavelab and reproducible research. In Wavelets and statistics (Springer), pp. 55–81.

Business Intelligence. (2004). Pilot study: optimum refresh cycle and method for desktop outsourcing, Technical Report (Intel Business Center).

Carcea, I., Caraballo, N.L., Marlin, B.J., Ooyama, R., Navarro, J.M.M., Opendak, M., Diaz, V.E., Schuster, L., Torres, M.I.A., Lethin, H., et al. (2019). Oxytocin neurons enable social transmission of maternal behavior. Preprint at bioRxiv. https://doi.org/10.1101/845495.

Carpenter, A.E., Jones, T.R., Lamprecht, M.R., Clarke, C., Kang, I.H., Friman, O., Guertin, D.A., Chang, J.H., Lindquist, R.A., Moffat, J., et al. (2006). CellProfiler: image analysis software for identifying and quantifying cell phenotypes. Genome Biol. *7*, R100. https://doi.org/10.1186/gb-2006-7-10-r100.

Carver, J.C., Gesing, S., Katz, D.S., Ram, K., and Weber, N. (2018). Conceptualization of a us research software sustainability institute (URSSI). Comput. Sci. Eng. *20*, 4–9. https://doi.org/10.1109/MCSE.2018.03221924.

Chan Zuckerberg Initiative. (2019). Essential open source software for science (EOSS) - chan Zuckerberg initiative. https://chanzuckerberg.com/eoss/.

Chen, S., Loper, J., Chen, X., Zador, T., and Paninski, L. (2020). BAR code DEmixing through non-negative spatial Regression (BarDensr). Preprint at bioRxiv. https://doi.org/10.1371/journal.pcbi.1008256.

Chen, X., Dallmeier-Tiessen, S., Dasler, R., Feger, S., Fokianos, P., Gonzalez, J.B., Hirvonsalo, H., Kousidis, D., Lavasa, A., Mele, S., et al. (2019). Open is not enough. Nat. Phys. *15*, 113–119. https://doi.org/10.1038/s41567-018-0342-2.

Cheng, J., and Guo, J.L. (2018). How do the open source communities address usability and ux issues?: an exploratory study. In CHI EA '18: Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems, p. LBW523.

Couto, J., Musall, S., Sun, X.R., Khanal, A., Gluf, S., Saxena, S., Kinsella, I., Abe, T., Cunningham, J.P., Paninski, L., and Churchland, A.K. (2021). Chronic, cortex-wide imaging of specific cell populations during behavior. Nat. Protoc. *16*, 3241–3263. https://doi.org/10.1038/s41596-021-00527-z.

Crook, S.M., Davison, A.P., and Plesser, H.E. (2013). Learning from the past: approaches for reproducibility in computational neuroscience. Years of computational neuroscience, *Vol. 20* (Springer), pp. 73–102.

Dandi Team. (2019). Dandi Archive. https://www.dandiarchive.org/.

de Chaumont, F., Dallongeville, S., Chenouard, N., Hervé, N., Pop, S., Provoost, T., Meas-Yedid, V., Pankajakshan, P., Lecomte, T., Le Montagner, Y., et al. (2012). Icy: an open BioImage informatics platform for extended reproducible research. Nat. Methods *9*, 690–696. https://doi.org/10.1038/nmeth.2075.

Demchenko, Y., Grosso, P., Laat, C.D., and Membrey, P. (2013). Addressing big data issues in scientific data infrastructure. In International Conference on Collaboration Technologies and Systems (CTS), pp. 48–55.

Dietterich, T.G. (2000). Multiple classifier systems. In First International Workshop, MCS 2000 Cagliari, Italy, June 21-23, 2000 Proceedings, pp. 1–15.

Donoho, D.L. (2010). An invitation to reproducible computational research. Biostatistics *11*, 385–388. https://doi.org/10.1093/biostatistics/kxq028.

Editorial. (2014). Code share: nature news and comment. https://www.nature.com/news/code-share-1.16232.

Flywheel Exchange. (2019). Flywheel: informatics platform for biomedical research and collaboration. https://flywheel.io/.

Fort, S., Hu, H., and Lakshminarayanan, B. (2019). Deep ensembles: a loss landscape perspective. Preprint at arXiv. arXiv:1912.02757.

Freeman, J. (2015). Open source tools for large-scale neuroscience. Curr. Opin. Neurobiol. *32*, 156–163. https://doi.org/10.1016/j.conb.2015.04.002.

Gao, Y., Archer, E.W., Paninski, L., and Cunningham, J.P. (2016). Linear dynamical neural population models through nonlinear embeddings. Adv. Neural Inf. Process. Syst. *29*, 163–171.

Ghosh, S.S., Poline, J.-B., Keator, D.B., Halchenko, Y.O., Thomas, A.G., Kessler, D.A., and Kennedy, D.N. (2017). A very simple, re-executable neuroimaging publication. F1000Res. *6*, 124.

Giovannucci, A., Friedrich, J., Gunn, P., Kalfon, J., Brown, B.L., Koay, S.A., Taxidis, J., Najafi, F., Gauthier, J.L., Zhou, P., et al. (2019). CaImAn an open source tool for scalable calcium imaging data analysis. eLife *8*, e38173. https://doi.org/10.7554/eLife.38173.

Giovannucci, A., Friedrich, J., Kaufman, M., Churchland, A., Chklovskii, D., Paninski, L., and Pnevmatikakis, E.A. (2017). Onacid: online analysis of calcium imaging data in real time. In Advances in Neural Information Processing Systems, pp. 2381–2391.

Glatard, T., Lewis, L.B., Ferreira da Silva, R., Adalat, R., Beck, N., Lepage, C., Rioux, P., Rousseau, M.-E., Sherif, T., Deelman, E., et al. (2015). Reproducibility of neuroimaging analyses across operating systems. Front. Neuroinfo. *9*, 12. https://doi.org/10.3389/fninf.2015.00012.

Goecks, J., Nekrutenko, A., Taylor, J., and Galaxy Team. (2010). Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. Genome Biol. *11*, R86. https://doi.org/10.1186/gb-2010-11-8-r86.

Goodman, D.F.M., and Brette, R. (2009). The brian simulator. Front. Neurosci. *3*, 192–197. https://doi.org/10.3389/neuro.01.026.2009.

Gorgolewski, K., Burns, C.D., Madison, C., Clark, D., Halchenko, Y.O., Waskom, M.L., and Ghosh, S.S. (2011). Nipype: a flexible, lightweight and extensible neuroimaging data processing framework in python. Front. Neuroinfo. *5*, 13. https://doi.org/10.3389/fninf.2011.00013.

Gorgolewski, K.J., Alfaro-Almagro, F., Auer, T., Bellec, P., Capotă, M., Chakravarty, M.M., Churchill, N.W., Cohen, A.L., Craddock, R.C., Devenyi, G.A., et al. (2017). BIDS apps: improving ease of use, accessibility, and reproducibility of neuroimaging data analysis methods. PLoS Comput. Biol. *13*, e1005209. https://doi.org/10.1371/journal.pcbi.1005209.

Graving, J.M., Chae, D., Naik, H., Li, L., Koger, B., Costelloe, B.R., and Couzin, I.D. (2019). DeepPoseKit, a software toolkit for fast and robust animal pose estimation using deep learning. eLife *8*, e47994. https://doi.org/10.7554/eLife.47994.

Hanson, B., Sugden, A., and Alberts, B. (2011). Making data maximally available. Science *331*, 649.

Hinsen, K. (2015). Technical debt in computational science. Comput. Sci. Eng. *17*, 103–107. https://doi.org/10.1109/MCSE.2015.113.

Hoffa, C., Mehta, G., Freeman, T., Deelman, E., Keahey, K., Berriman, B., and Good, J. (2008). On the use of cloud computing for scientific workflows. In IEEE fourth international conference on eScience, pp. 640–645.

Januszewski, M., Kornfeld, J., Li, P.H., Pope, A., Blakely, T., Lindsey, L., Maitin-Shepard, J., Tyka, M., Denk, W., and Jain, V. (2018). , High-precision automated reconstruction of neurons with flood-filling networks. Nat. Methods *15*, 605–610. https://doi.org/10.1038/s41592-018-0049-4.

Jararweh, Y., Al-Ayyoub, M., Darabseh, A., Benkhelifa, E., Vouk, M., and Rindos, A. (2016). Software defined cloud: survey, system and evaluation. Future Gener. Comput. Syst. *58*, 56–74. https://doi.org/10.1016/j.future.2015.10.015.

J.Gold Associates LLC (2014). Replacing enterprise PCs: the fallacy of the 3–4 year upgrade cycle, Technical Report (J.Gold Associates LLC).

Kane, G.A., Lopes, G., Saunders, J.L., Mathis, A., and Mathis, M.W. (2020). Real-time, low-latency closed-loop feedback using markerless posture tracking. eLife *9*, e61909. https://doi.org/10.7554/eLife.61909.

Köster, J., and Rahmann, S. (2012). Snakemake—a scalable bioinformatics workflow engine. Bioinformatics *28*, 2520–2522. https://doi.org/10.1093/bioinformatics/bts480.

Krafczyk, M., Shi, A., Bhaskar, A., Marinov, D., and Stodden, V. (2019). Scientific tests and continuous integration strategies to enhance reproducibility in the scientific software context. In Proceedings of the 2nd International Workshop on Practical Reproducible Evaluation of Computer Systems, pp. 23–28.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. In 31st Conference on Neural Information Processing Systems (NIPS 2017).

Landhuis, E. (2017). Neuroscience: big brain, big data. Nature *541*, 559–561. https://doi.org/10.1038/541559a.

Lee, J.H., Carlson, D.E., Razaghi, H.S., Yao, W., Goetz, G.A., Hagen, E., Batty, E., Chichilnisky, E., Einevoll, G.T., and Paninski, L. (2017). Yass: yet another spike sorter. In Advances in Neural Information Processing Systems 30 (NIPS 2017), pp. 4002–4012.

Lopes, G., Bonacchi, N., Frazão, J., Neto, J.P., Atallah, B.V., Soares, S., Moreira, L., Matias, S., Itskov, P.M., Correia, P.A., et al. (2015). Bonsai: an event-based framework for processing and controlling data streams. Front. Neuroinform. *9*, 7. https://doi.org/10.3389/fninf.2015.00007.

Magland, J., Jun, J.J., Lovero, E., Morley, A.J., Hurwitz, C.L., Buccino, A.P., Garcia, S., and Barnett, A.H. (2020). SpikeForest, reproducible web-facing ground-truth validation of automated neural spike sorters. eLife *9*, e55167. https://doi.org/10.7554/eLife.55167.

Mahvi, J., and Zarfaty, A. (2009). Using TCO to Determine PC Upgrade Cycles (Intel Corporation).

Mathis, A., Mamidanna, P., Cury, K.M., Abe, T., Murthy, V.N., Mathis, M.W., and Bethge, M. (2018). DeepLabCut: markerless pose estimation of user-defined body parts with deep learning. Nat. Neurosci. *21*, 1281–1289. https://doi.org/10.1038/s41593-018-0209-y.

Merali, Z. (2010). Computational science: error. Nature *467*, 775–777.

Merkel, D. (2014). Docker: lightweight linux containers for consistent development and deployment. Linux J. *239*, 2.

Miller, G. (2006). Scientific publishing. A scientist's nightmare: software problem leads to five retractions. Science *314*, 1856–1857. https://doi.org/10.1126/science.314.5807.1856.

Minka, T.P. (1999). From hidden markov models to linear dynamical systems. Technical report (MIT).

Monajemi, H., Murri, R., Jonas, E., Liang, P., Stodden, V., and Donoho, D.L. (2019). Ambitious data science can be painless. Preprint at arXiv. arXiv:1901.08705.

Morey, T., and Nambiar, R. (2009). Using total cost of owner-ship to determine optimal PC refresh lifecycles. Technical Report (Wipro Limited Company).

Morris, K. (2016). Infrastructure as Code: Managing Servers in the Cloud (O'Reilly Media, Inc.).

Musall, S., Kaufman, M.T., Juavinett, A.L., Gluf, S., and Churchland, A.K. (2019). Single-trial neural dynamics are dominated by richly varied movements. Nat. Neurosci. *22*, 1677–1686. https://doi.org/10.1038/s41593-019-0502-4.

Neuro Cloud Consortium. (2016). To the cloud! A grassroots proposal to accelerate brain science discovery. Neuron *92*, 622–627. https://doi.org/10.1016/j.neuron.2016.10.033.

NeuroScout. (2022). Neuroscout. https://neuroscout.github.io/neuroscout.

Nichols, D.M., Thomson, K., and Yeates, S.A. (2001). Usability and open-source software development. In CHINZ01, pp. 49–54.

Nilsson, S.R., Goodwin, N.L., Choong, J.J., Hwang, S., Wright, H.R., Norville, Z.C., Tong, X., Lin, D., Bentzley, B.S., Eshel, N., et al. (2020). Simple Behavioral Analysis (SimBA) – an open source toolkit for computer classification of complex social behaviors in experimental animals. Preprint at bioRxiv. https://doi.org/10.1101/2020.04.19.049452.

Nowogrodzki, A. (2019). How to support open source software and stay sane. Nature *571*, 133–134. https://doi.org/10.1038/d41586-019-02046-0.

Ovadia, Y., Fertig, E., Ren, J., Nado, Z., Sculley, D., Nowozin, S., Dillon, J., Lakshminarayanan, B., and Snoek, J. (2019). Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In Advances in Neural Information Processing Systems, p. 32.

Pachitariu, M., Steinmetz, N.A., Kadir, S.N., Carandini, M., and Harris, K.D. (2016). Fast and accurate spike sorting of high-channel count probes with KiloSort. In Advances in Neural Information Processing Systems, pp. 4448–4456.

Pachitariu, M., Stringer, C., Dipoppa, M., Schroder, S., Rossi, L.F., Dalgleish, H., Carandini, M., and Harris, K.D. (2017). Suite2p: beyond 10,000 neurons with standard two-photon microscopy. Preprint at bioRxiv. 061507. https://doi.org/10.1101/061507.

Pandarinath, C., O'Shea, D.J., Collins, J., Jozefowicz, R., Stavisky, S.D., Kao, J.C., Trautmann, E.M., Kaufman, M.T., Ryu, S.I., Hochberg, L.R., et al. (2018). Inferring single-trial neural population dynamics using sequential auto-encoders. Nat. Methods *15*, 805–815. https://doi.org/10.1038/s41592-018-0109-9.

Paninski, L., and Cunningham, J.P. (2018). Neural data science: accelerating the experiment-analysis-theory cycle in large-scale neuroscience. Curr. Opin. Neurobiol. *50*, 232–241. https://doi.org/10.1016/j.conb.2018.04.007.

Parthasarathy, N., Batty, E., Falcon, W., Rutten, T., Rajpal, M., Chichilnisky, E., and Paninski, L. (2017). Neural networks for efficient bayesian decoding of natural images from retinal neurons. In Advances in Neural Information Processing Systems, pp. 6434–6445.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). PyTorch: an imperative style, high-performance deep learning library. In Advances in Neural Information Processing Systems, pp. 8024–8035.

Pnevmatikakis, E.A., Soudry, D., Gao, Y., Machado, T.A., Merel, J., Pfau, D., Reardon, T., Mu, Y., Lacefield, C., Yang, W., et al. (2016). Simultaneous denoising, deconvolution, and demixing of calcium imaging data. Neuron *89*, 285–299. https://doi.org/10.1016/j.neuron.2015.11.037.

Radiuk, P.M. (2017). Impact of training set batch size on the performance of convolutional neural networks for diverse datasets. Inf. Technol. Manag. Sci. *20*, 20–24. https://doi.org/10.1515/itms-2017-0003.

Raff, E. (2019). A step toward quantifying independently reproducible machine learning research. In Advances in Neural Information Processing Systems, p. 32.

Riley, J. (2010). Starcluster-numpy/scipy computing on amazon's elastic compute cloud (ec2). In SciPY2010: python for scientific computing conference (Austin).

Rocklin, M. (2015). Dask: parallel computation with Blocked algorithms and task scheduling. In Proceedings of the 14th Python in Science Conference, pp. 130–136.

Rokem, A., Dichter, B., Holdgraf, C., and Ghosh, S. (2021). Pan-neuro: interactive computing at scale with brain datasets (OSF Preprints). https://osf.io/mwh2b.

Rübel, O., Tritt, A., Dichter, B., Braun, T., Cain, N., Clack, N., Davidson, T., Dougherty, M., Fillion-Robin, J.-C., Graddis, N., et al. (2019). Nwb: N 2.0: an accessible data standard for neurophysiology. Preprint at bioRxiv. https://doi.org/10.1101/523035.

Rübel, O., Tritt, A., Ly, R., Dichter, B.K., Ghosh, S., Niu, L., Soltesz, I., Svoboda, K., Frank, L., and Bouchard, K.E. (2021). The neurodata without borders ecosystem for neurophysiological data science. Preprint at bioRxiv. https://doi.org/10.1101/2021.03.13.435173.

Sanielevici, S., Sivagnanam, S., Yoshimoto, K., Carnevale, N.T., and Majumdar, A. (2018). The neuroscience Gateway: enabling large scale modeling and data processing in neuroscience. In PEARC '18: Proceedings of the Practice and Experience on Advanced Research Computing, p. 52.

Saxena, S., Kinsella, I., Musall, S., Kim, S.H., Meszaros, J., Thibodeaux, D.N., Kim, C., Cunningham, J., Hillman, E.M.C., Churchland, A., and Paninski, L. (2020). Localized semi-nonnegative matrix factorization (LocaNMF) of wide-field calcium imaging data. PLoS Comput. Biol. *16*, e1007791. https://doi.org/10.1371/journal.pcbi.1007791.

Seven Bridges Genomics. (2019). The seven bridges platform. https://www.sbgenomics.com/.

Schneider, C.A., Rasband, W.S., and Eliceiri, K.W. (2012). NIH Image to ImageJ: 25 years of image analysis. Nat. Methods *9*, 671–675. https://doi.org/10.1038/nmeth.2089.

Schweihoff, J.F., Loshakov, M., Pavlova, I., Kück, L., Ewell, L.A., and Schwarz, M.K. (2021). Deeplabstream enables closed-loop behavioral experiments using deep learning-based markerless, real-time posture detection. Commun. Biol. *4*, 130. https://doi.org/10.1038/s42003-021-01654-9.

Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., Chaudhary, V., Young, M., Crespo, J.-F., and Dennison, D. (2015). Hidden technical debt in machine learning systems. In Advances in Neural Information Processing Systems, p. 28.

Simonyan, V., and Mazumder, R. (2014). High-performance integrated virtual environment (hive) tools and applications for big data analysis. Genes *5*, 957–981. https://doi.org/10.3390/genes5040957.

Sommer, C., Straehle, C., Koethe, U., and Hamprecht, F.A. (2011). Ilastik: interactive learning and segmentation toolkit. In IEEE International Symposium on Biomedical Imaging: From Nano to Macro, pp. 230–233.

Steinmetz, N.A., Aydin, C., Lebedeva, A., Okun, M., Pachitariu, M., Bauza, M., Beau, M., Bhagat, J., Böhm, C., Broux, M., et al. (2021). Neuropixels 2.0: a miniaturized high-density probe for stable, long-term brain recordings. Science *372*, eabf4588. https://doi.org/10.1126/science.abf4588.

Stodden, V., Seiler, J., and Ma, Z. (2018). An empirical analysis of journal policy effectiveness for computational reproducibility. Proc. Natl. Acad. Sci. USA *115*, 2584–2589. https://doi.org/10.1073/pnas.1708290115.

Sussillo, D., Jozefowicz, R., Abbott, L.F., and Pandarinath, C. (2016). LFADS—latent factor analysis via dynamical systems. Preprint at arXiv. arXiv:1608.06315.

Teeters, J.L., Godfrey, K., Young, R., Dang, C., Friedsam, C., Wark, B., Asari, H., Peron, S., Li, N., Peyrache, A., et al. (2015). Neurodata without borders: creating a common data format for neurophysiology. Neuron *88*, 629–634. https://doi.org/10.1016/j.neuron.2015.10.025.

Terra. (2022). Terra. https://app.terra.bio.

Terry, M., Kay, M., and Lafreniere, B. (2010). Perceptions and practices of usability in the free/open source software (FoSS) community. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 999–1008.

Towns, J., Peterson, G.D., Roskies, R., Scott, J.R., Wilkins-Diehr, N., Cockerill, T., Dahan, M., Foster, I., Gaither, K., Grimshaw, A.G.D., et al. (2014). XSEDE: accelerating scientific discovery. Comput. Sci. Eng. *16*, 62–74. https://doi.org/10.1109/MCSE.2014.80.

Tukey, J.W. (1962). The future of data analysis. Ann. Math. Statist. *33*, 1–67. https://doi.org/10.1214/aoms/1177704711.

Waltz, D., and Buchanan, B.G. (2009). Computer science. Automating science. Science *324*, 43–44. https://doi.org/10.1126/science.1172781.

Whiteway, M.R., Biderman, D., Friedman, Y., Dipoppa, M., Buchanan, E.K., Wu, A., Zhou, J., Noel, J.-P., The International Brain Laboratory, Cunningham, J., and Paninski, L. (2021). Partitioning variability in animal behavioral videos using semi-supervised variational autoencoders. Preprint at bioRxiv. https://doi.org/10.1371/journal.pcbi.1009439.

Wiltschko, A.B., Johnson, M.J., Iurilli, G., Peterson, R.E., Katon, J.M., Pashkovski, S.L., Abraira, V.E., Adams, R.P., and Datta, S.R. (2015). Mapping sub-second structure in mouse behavior. Neuron *88*, 1121–1135. https://doi.org/10.1016/j.neuron.2015.11.031.

Wu, A., Buchanan, E.K., Whiteway, M.R., Schartner, M., Meijer, G., Noel, J.-P., Rodriguez, E., Everett, C., Norovich, A., Schaffer, E., et al. (2020). Deep Graph Pose: a semi-supervised deep graphical model for improved animal pose tracking. Preprint at bioRxiv. https://doi.org/10.1101/2020.08.20.259705.

Yatsenko, D., Reimer, J., Ecker, A.S., Walker, E.Y., Sinz, F., Berens, P., Hoenselaar, A., Cotton, R.J., Siapas, A.S., and Tolias, A.S. (2015). DataJoint: managing big scientific data using MATLAB or Python. Preprint at bioRxiv. https://doi.org/10.1101/031658.

Yoo, A.B., Jette, M.A., and Grondona, M. (2003). Job scheduling strategies for parallel processing. In Job Scheduling Strategies for Parallel Processing: 9th International Workshop, JSSPP 2003, pp. 44–60.

Yu, B.M., Cunningham, J.P., Santhanam, G., Ryu, S.I., Shenoy, K.V., and Sahani, M. (2009). Gaussian-process factor analysis for low-dimensional single-trial analysis of neural population activity. J. Neurophysiol. *102*, 614–635. https://doi.org/10.1152/jn.90941.2008.

Zhao, L., and Deek, F.P. (2005). Improving open source software usability. In AMCIS 2005 Proc., p. 430.

Zhou, A., He, B., Ibrahim, S., Buyya, R., Calheiros, R., and Dastjerdi, A. (2016). eScience and big data workflow in clouds: a taxonomy and survey. In Big data: Principles and Paradigms (O'Reilly Media, Inc.), pp. 431–456.

Zhou, P., Resendez, S.L., Rodriguez-Romaguera, J., Jimenez, J.C., Neufeld, S.Q., Giovannucci, A., Friedrich, J., Pnevmatikakis, E.A., Stuber, G.D., Hen, R., et al. (2018). Efficient and accurate extraction of in vivo calcium signals from microendoscopic video data. eLife *7*, e28728. https://doi.org/10.7554/eLife.28728.

STAR★METHODS

KEY RESOURCES TABLE

REAGENT or RESOURCE	SOURCE	IDENTIFIER
Deposited data		
Raw data used for the benchmarking of CaImAn (Giovannucci et al., 2019)	Zenodo	Zenodo: https://doi.org/10.5281/zenodo.1659149
Performance quantification data used to report timing and cost of analyses on NeuroCAAS (related to Table 1 and Figure 8)	Zenodo	CSHL Scientific Digital Repository: https://doi.org/10.5281/zenodo.6512118
Raw data used for to test WFCI analysis.	Cold Spring Harbor Repository	Zenodo: https://doi.org/10.14224/1.38599
Software and algorithms		
Source repository used to build analyses from blueprints.	Zenodo	Zenodo: https://doi.org/10.5281/zenodo.6512118
Contributor repository used to help developers add analyses to NeuroCAAS.	Zenodo	Zenodo: https://doi.org/10.5281/zenodo.6512121
Interface repository used to build the website www.neurocaas.org	Zenodo	Zenodo: https://doi.org/10.5281/zenodo.6512125
Repository used to generate ensemble outputs from individually trained models	Zenodo	Zenodo: https://doi.org/10.5281/zenodo.6513057

RESOURCE AVAILABILITY

Lead contact

Further information and requests for resources should be directed to and will be fulfilled by the lead contact, John P. Cunningham (jpc2181@columbia.edu).

Materials availability

This study did not generate new unique reagents.

Data and code availability

- Quantifications of performance and reproducibility of NeuroCAAS have been deposited at Zenodo and are publicly available as of the date of publication. DOIs are listed in the key resources table.
- This paper analyzes existing, publicly available data. These accession numbers for the datasets are listed in the key resources table.
- All other data reported in this paper will be shared by the lead contact upon request.
- All original code has been deposited at Zenodo and is publicly available as of the date of publication. DOIs are listed in the key resources table.

METHOD DETAILS

NeuroCAAS architecture specifics

The software supporting the NeuroCAAS platform has been divided into three separate Github repositories. The first, https://github.com/cunningham-lab/neurocaas is the main repository that hosts the Infrastructure-as-Code implementation of NeuroCAAS. We will refer to this repository as the source repo throughout this section. The source repo is supported by two additional repositories: https://github.com/cunningham-lab/neurocaas_contrib hosts contribution tools to assist in the development and creation of new analyses on NeuroCAAS, and https://github.com/jjhbriggs/neurocaas_frontend hosts the website interface to NeuroCAAS. We will refer to these as the contrib repo and the interface repo respectively throughout this section. We discuss the relationship between these repositories in the following section, and in Figure S5.

Source repo

Section NeuroCAAS builds complete infrastructure stacks gives an overview of how NeuroCAAS encodes individual analyses into blueprints, and deploys them into full infrastructure stacks, following the principle of Infrastructure-as-Code (IaC). This section presents blueprints in more depth and show how the whole NeuroCAAS platform can be managed through IaC, encoding features such as user data storage, credentials, and logging infrastructure in code documents analogous to analysis blueprints as well. All of these code documents, together with code to deploy them, make up NeuroCAAS′s source repo. There is a one-to-one correspondence between NeuroCAAS′s source repo and infrastructure components: deploying the source repo provides total coverage of all the infrastructure needed to analyze data on NeuroCAAS (Figure S5, bottom). Although much of the code to translate blueprints and other infrastructure code necessarily references AWS resources, NeuroCAAS blueprints and other IaC artefacts are not tied to AWS, except in their reliance on particular hardware instance configurations. We can potentially recreate these hardware instances in other public clouds, using existing tools to support cloud-agnostic IaC approaches, as suggested by Brikman (2019). Doing so will further improve the scale and robustness of our platform.

Within the source repo, each NeuroCAAS blueprint (see Figure S2 for an example) is formatted as a JSON document with predefined fields. The expected values for most of these fields identify a particular cloud resource, such as the ID for an immutable analysis environment, or a hardware identifier to specify an instance within the resource bank (Lambda.LambdaConfig.AMI and Lambda.LambdaConfig.INSTANCE_TYPE in Figure S2, respectively). Upon deployment, these fields determine the creation of certain cloud resources: AWS EC2 Amazon Machine Images in the case of IAE IDs, and AWS EC2 Instances in the case of hardware identifiers. One notable exception is the protocol specifying behavior of a corresponding NeuroCAAS job manager (Lambda.CodeUri and Lambda.Handler in Figure S2). Instead of identifying a particular cloud resource, each blueprint's protocol is a python module within the source repo that contains functions to execute tasks on the cloud in response to user input. The ability to specify protocols in python allows NeuroCAAS to support the complex workflows shown in Figure 5. Job managers are deployed from these protocols as AWS Lambda functions that execute the protocol code for a particular analysis whenever users submit data and parameters. Since all parts of NeuroCAAS workflow can be managed with python code (i.e. through a programmatic interface, job manager protocol, or within the IAE itself), external workflow management tools can easily be integrated to analyses on a case-by-case basis in order to deploy the scale of NeuroCAAS in parallel or sequentially, as needed.

Another major aspect of NeuroCAAS′s source repo that is not discussed in the results is the management of individual users. NeuroCAAS applies the same IaC principles to user creation and management as it does to individual analyses. To add a new user to the platform, NeuroCAAS first creates a corresponding user profile in the source repo (Figure S5, right), that specifies user budgets, creates private data storage space, generates their (encrypted) security credentials, and identifies other users who they collaborate with. Users resources are created using the AWS Identity and Accesss Management (IAM) service.

Contrib and interface repos

Given only the NeuroCAAS source repo, analyses can be hosted on the NeuroCAAS platform and new users can be added to the platform simply by deploying the relevant code documents. However, interacting directly with resources provided by the NeuroCAAS source repo can be challenging for both analysis users and developers. For developers, the steps required to fill in a new analysis blueprint may not be clear, and the scripting steps necessary within an IAE to retrieve user data and parameters requires knowledge of specific resources on the Amazon Web Services cloud. For users, the NeuroCAAS source repo on its own does not support an intuitive interface or analysis documentation, requiring users to interact with NeuroCAAS through generic cloud storage browsers, forcing them to engage in tedious tasks like navigating file storage and downloading logs before examining them. Collectively, these tasks lower the accessibility that is a key part of NeuroCAAS 's intended design. To handle these challenges, we created two additional code repositories, the NeuroCAAS contrib repo and interface repo, for developers and users, respectively.

The NeuroCAAS contrib repo supports a command line tool and python code to streamline the process of developing and creating new NeuroCAAS analyses. During the development process, the NeuroCAAS contrib repo can create infrastructure stacks independently of input-triggered job managers for a limited time, allowing developers to build and test IAEs interactively on powerful hardware instances in ‘‘debug mode’’ (Figure S5, bottom right), and populate the analysis blueprint as they go. Then, when a new analysis is ready to be used on NeuroCAAS, the NeuroCAAS contrib repo automatically versions the entire source repository after integrating and deploying the new blueprint, generating a unique analysis version ID. All NeuroCAAS analyses can be updated only by directly editing blueprints, and blueprints are assigned a new analysis version ID every time that they are updated. By enforcing a tight correspondence between blueprints and analyses, we ensured the reproducibility of all analyses conducted via NeuroCAAS, regardless of ongoing updates to the underlying infrastructure or algorithm (Figure S5, top right). With an analysis version ID, it is possible to replicate results that were generated with older versions of some analysis algorithm, making this a particularly useful feature for users processing data with an analysis that is still actively being developed. The NeuroCAAS developer documentation linked in additional resources contains a detailed guide for developers to get started with NeuroCAAS.

The NeuroCAAS interface repo supports the website interface to NeuroCAAS, hosted at www.neurocaas.org. In addition to providing documentation and a simpler user interface, (Figure S5, bottom left) the interface repo interacts with the source repo to automatically create and deploy user profiles when users sign up, significantly increasing the potential scale of the platform (Figure S5, top left). This website based user credentialing system can be referenced by other user interfaces as well, as is done in https://github.com/jcouto/wfield. If users wish to share analysis access and data with other users, they can also use the website to create and request unique ‘‘group codes’’ at sign up, that they can use to invite other users into the same group. Doing so allows them to easily share analysis access with others.

Developer workflow

In this section we give more specific steps of how developers and authors built analyses for NeuroCAAS. See also Figure S6 for a corresponding schematic.

Flexible installation and scripting

Developers first install their core analysis into an IAE and hardware instance, just as they would with a local computer. Within an IAE any programming language can be supported, although certain precautions must be taken when working with licensed software such as Matlab (see developer documentation for more details https://neurocaas.readthedocs.io/en/latest/index.html). We provide tools to help developers write a workflow script to make their core analysis immutable (Figure S6). Post installation, the configured IAE is automatically saved in a blueprint (Figure S6, right).

Simple input/output handling

All NeuroCAAS analyses take a dataset file and a YAML formatted configuration file as input. Datasets can be parallelized over, and configuration files can specify any kind of parameter, including paths to supplemental data files (we provide an example of such a workflow in our custom analyses). NeuroCAAS handles transfer of data files from users into an IAE, and likewise writes any results back to users in timestamped folders that ensure version control for analysis outputs. All messages that would be printed to the IAE console during analysis are delivered back to the user in real time as a set of automatically formatted log files, along with job success/ failure status messages, CPU usage, and memory usage. Developers can specify any files to be returned to the user, including custom logs, or intermediate results that would be useful to examine as analysis proceeds.

Testing with private data storage

Each user of a given analysis has a password protected account that authorizes them to interface with their own private cloud storage. Users have separated input and output areas in cloud storage, where they can maintain datasets for re-analysis, or keep intermediate analysis results as convenient. Although we have not capped the size of data we allow each user to store overall, we restrict both as follows: users cannot download data from input areas (although they can delete), and they cannot upload to output areas. These restrictions have distinct benefits for cost and reproducibility. In the later stages of testing, developers can upload test data and configuration files to private data storage exactly as a user would, and ensure that results and logging information appear as intended before releasing their analysis to the public. Developers can also set up selective access for designated test users before releasing the analysis to the general public. At this stage, we also work with developers to determine the optimal hardware instance type from the resource bank for their analysis and to determine if additional configuration of a custom job manager is necessary. We note, importantly that our current storage solution does not meet HIPAA standards and should not be used for sensitive health records.

Reproducible use and development

By default, each analysis on NeuroCAAS provisions a single type of hardware instance for all data. However, if needed instances can be provisioned in a dataset-dependent manner, adjusting the size of storage volumes, memory, or other computing resources. These per-job changes are still recorded in versioned logs to ensure reproducibility of all jobs (see immutable analysis environments for software infrastructure for details). Such dataset dependent changes can be triggered by users, with the CLI interface, or programmed by developers through the job manager. In cases where users report bugs, analysis developers can then access the exact same IAE, resource bank instance, and inputs in interactive "debug mode" once again, making changes and redeploying the blueprint exactly as they did in the initial deployment. Furthermore, if developers would like to continue updating their analysis, they can do so without impacting the reproducibility of existing results, because each NeuroCAAS job produces an analysis ID identifying a particular blueprint version.

Novel analyses

For each novel analysis described in "NeuroCAAS simplifies large data pipelines: widefield imaging protocol" and "NeuroCAAS stabilizes deep learning models: ensemble markerless tracking," we provide details on its component infrastructure stacks, as well as details on relevant development outside the NeuroCAAS framework we have already presented.

Widefield imaging

The Widefield Calcium Imaging analysis that we present involves two independent infrastructure stacks, with the second taking as input the results of the first. The first infrastructure stack performs motion correction, denoising, compression, and hemodynamic correction, and is performed on an instance with 64 virtual cores (further infrastructure details are identical to the "PMD" row of Table S5). The second infrastructure stack performs demixing of denoised, corrected widefield imaging data, and is performed on an instance with a Tesla V100 GPU (further infrastructure details are identical to the "LocaNMF" row of Table S5). In addition to these two infrastructure stacks, we support a custom GUI (available at https://github.com/jcouto/wfield). This user interface integrates with the credentials generated for users on the NeuroCAAS website, allowing users who have signed up via the website to use the GUI with an existing account. The GUI hosts a number of initialization steps on the user's local machine, involving selection of parameters and alignment of data to landmarks on a given brain atlas. The GUI is also able to upload data directly to NeuroCAAS cloud storage, submit jobs, and monitor their progress. Next, the GUI is able to detect when the first step of processing is completed, and submits the relevant results files as input to the second step, mimicking the steps a user would take manually to manage this process. Finally, when all processing is complete the GUI retrieves analysis results back to the user's local machine. For more details on implementation of each analysis step, please see Couto et al. (2021).

Ensemble markerless tracking

The deep ensembling analysis that we present is also performed in two separate infrastructure stacks, but both the initial training and the consensus output generation steps are performed on the same IAE and resource bank instance. In both cases, we use an instance equipped with a Tesla V100 GPU, otherwise identical to the infrastructure shown in the DeepLabCut row of Table S5. We trained DeepGraphPose with the default training settings provided in the file run_dgp_demo.py within the core DeepGraphPose analysis code, on the "twomice-top-down" data from the DeepGraphPose paper (Wu et al., 2020). That paper provides full videos of analysis of this dataset using a single DeepGraphPose model. To enable ensembling, we built a separate set of ensembling tools that work with DeepGraphPose (Wu et al., 2020) - they can be found at https://github.com/cunningham-lab/neurocaas_ensembles. In order to create a consensus output, we averaged the confidence maps from each model in an ensemble in the following way: Assume a set of N trained DGP networks, $\varphi_i, i \in 1\ldots N$, and a video frame, $F \in \mathbb{R}^{X\times Y\times 3}$ with three color channels. Assume that the network has been trained to track a single body part (the general case follows immediately), and take the scoremap outputs (unnormalized likelihoods) on this image from the output convolutional layer, denoted $\varphi_i^{sc}(F)$, where each scoremap $\varphi_i^{sc}(F) \in \mathbb{R}^{X\times Y}$. These scoremap outputs are unnormalized likelihoods representing the probability that the body part of interest is located in any individual pixel of the image. Then, we can compute the mean scoremap for a given image as:

$$\overline{\varphi}^{sc}(F) = S^{-1}\left(\frac{1}{N}\sum_i S\left(\varphi_i^{sc}(F)\right)\right) \qquad \text{(Equation 1)}$$

Where S is the elementwise sigmoid function. The consensus output is then calculated from the softargmax function of this mean scoremap.

Furthermore, to calculate the rmse error, we use the following metric: Assume we have detections for all of the test frames in a video as a tensor, $x \in \mathbb{R}^{T\times D\times C}$, with entries x_{tdc}, where t represents the frame index, d the part index, and c the coordinate $\in [x,y]$. Likewise, we have groundtruth data g with entries g_{tdc} of the same dimension. Then the error is calculated as follows:

$$RMSE(x,g) = \sqrt{\frac{\sum_{t,d,c}\left[(x_{tdc} - g_{tdc})^2\right]}{T}} \qquad \text{(Equation 2)}$$

Details and implementation can be found in the repository https://github.com/cunningham-lab/neurocaas_ensembles, and the full analysis is available for use at http://neurocaas.org/analysis/14.

QUANTIFICATION AND STATISTICAL ANALYSIS

Quantifying reproducibility on NeuroCAAS

In order to quantify the reproducibility of analyses on NeuroCAAS, we selected two analyses already available on NeuroCAAS; CaImAn (Giovannucci et al., 2019), and Ensemble DeepGraphPose (NeuroCAAS stabilizes deep learning models: ensemble markerless tracking). We fixed in place the blueprint version for these analyses, as well as a dataset and configuration file, and compared the results of 15 independent runs for each of these analyses. These runs capture a variety of different real world interventions that can affect analysis reproducibility in practice. In particular, Runs 1-5 were performed by a paper author in the United States, Runs 6-10 were performed by a non-author researcher in India, and Runs 11-14 were performed by a non-author researcher in Switzerland. Within each of these sets of runs, we can test for the variation in analysis outputs over analysis runs conducted by the same researcher. Across these sets of runs, we can test for variation in analysis outputs over the physical location of the researcher performing experiments. Finally, for both analyses Run 15 was performed using an entirely separate instantiation of the NeuroCAAS platform, as described in the discussion, and for which detailed instructions are provided in our developer documentation. This run ensures that there is no dependence of our analysis results on the particular set of infrastructure resources used to build and implement NeuroCAAS analyses blueprints.

In the absence of a generic metric to compare analysis outputs, we chose specific metrics for each analysis.

CaImAn

We benchmarked CaImAn on the YST dataset introduced in the CaImAn paper (Giovannucci et al., 2019). To compare the outputs of CaImAn, we independently looked at the differences between the spatial components (referred to as the "A" matrix in the CaImAn software package) and the temporal components (referred to as the "C" matrix in the CaImAn software package) found by the system independently. For both temporal and spatial components, we assumed a deterministic ordering of components across runs. We compared components pairwise, and reported average quantities across paired components. To determine the variation of spatial components across runs, we quantified the Jaccard similarity coefficient pairwise between detected components, as has been done previously in Giovannucci et al. (2017). In order to compare the variation of temporal components across runs, we quantified the root mean square error between paired temporal components. In both cases, we found no detectable variation across runs.

Ensemble DeepGraphPose

We benchmarked Ensemble DeepGraphPose on the "twomice-top-down" data described in the methods above with the introduction of Ensemble Markerless Tracking. We compared the outputs of each run as the time series describing the temporal evolution of each individual body part. We quantified differences between two runs, x_1, x_2 as the root mean squared error between time series describing the evolution of each individual body part, averaged overall body parts. RMSE has previously been used in the literature to quantify the similarity between the tracked positions of behavioral markers, as in Mathis et al. (2018) and Wu et al. (2020). We expect this variation to come from non-deterministic computations used to speed up the computation of convolutions employed in the relevant behavioral tracking model.

Timings

As a separate but related question, we also quantified the variation in the time taken to analyze data on NeuroCAAS across these same 15 runs. We quantified timings as follows:

- The **Setup Time** of an analysis was calculated as the time between the moment when a job was submitted to user storage (measured as the upload timestamp associated with a job's "submit.json" file) and when the job was marked as started in the corresponding "certificate.txt" log in user storage (reported as a delta following the statement "JOB MONITOR LOG COMPLETE".)
- The **Analysis Time** was calculated as the time between the end of a job's Setup Time and the time when the last analysis output of a particular run is uploaded to user storage (measured using upload timestamps of those analysis results).
- The **Shutdown Time** of an analysis is the time between the end of a job's Analysis Time and the time when all resources associated with the job have been stopped. It is measured based upon the time when a job end marker file is written to user storage (called "end.txt").

The total timing of an analysis job is the sum of these three contributions. In practice, we care most about variation in the first two components, as these are most relevant to a user's experience of NeuroCAAS.

Quantifying usage

By default, NeuroCAAS records metadata for each job that is requested, such as the requester of the job, the time at which it was requested, and the datasets and config file that were analyzed. These quantities are necessary to enable per-user budgeting.

We aggregated this per-user metadata in order to generate the usage statistics shown in Figure 3. Aggregation and analysis of user data is now offered to developers through the contrib repo described above, and the data for Figure 3 panels were generated by running the command:

```
$ neurocaas-contrib monitor visualize-parallelism
```

This command is included with the developer CLI, which lists the usage of a single analysis bucket. We ran this command for the following analyses:

- epi-ncap-web
- dlc-ncap-web
- pmd-ncap-web
- caiman-ncap-web
- locanmf-ncap-web
- bardensr
- dlc-ncap-stable
- caiman-ncap-stable
- polleuxmonitored
- carceamonitored
- wfield-preprocess
- yass-ncap-stable
- one-photon-compress
- one-photon-demix
- one-photon-mcorr
- dgp-refactor
- ensemble-dgp
- label-job-create-web

We then iterated through all of these logs, and aggregated information across all platform usage for these analyses, using metadata for individual jobs to group results by user, by parallelism, and by developer. For each aggregated collection of jobs, we extracted the total corresponding number of compute hours as well, using the script figures/parallelized.py included in the contrib repo. Critically, here we exclude usage by NeuroCAAS team members testing analyses for deployment, which is run from the user accounts "reviewers","debuggers", and "examplegroup2". In rare cases where metadata logging failed, we excluded the corresponding analysis job from this quantification.

Benchmarking algorithms on NeuroCAAS

For each analysis currently on NeuroCAAS, the specific infrastructure choices in the corresponding blueprint (Figure S5, right) are given in Table S5. To meaningfully benchmark NeuroCAAS against current standards, we simulated corresponding local infrastructure. Local infrastructure was also built on AWS, and spans resources comparable to personal hardware and cluster compute, depending on the use case (see Table S6). As a general guideline, we chose local infrastructure representatives that would reasonably be available to a typical researcher, unless the datasets we considered required more powerful resources. To account for the diversity of resources available to neuroscience users, we offer alternative quantifications to those presented in Figure 7 in the supplemental information (see Figure S7), and make performance quantification data and calculations available to users who would like to compare to their own infrastructure through a custom tool on our project repository (see README: https://github.com/cunningham-lab/neurocaas).

For each analysis that we benchmarked on NeuroCAAS, we chose three datasets of increasing size as representative use cases of the algorithms in question. The size differences of these datasets reflect the diversity of potential use cases among different users of the same algorithm. The CaImAn benchmarking data consists of datasets N.02.00, J_123, J_115 from the data shared with the CaImAn paper (Giovannucci et al., 2019). Benchmark analysis is based on a script provided to regenerate Figure 7 of the CaImAn paper. Note that although this data could be batched, we choose to maintain all three datasets as contiguous wholes. We took advantage of the fact that the algorithm was built to parallelize across multiple cores of the same machine, and chose hardware to make effective use of this implementation across data sizes (for details see Figure 8 in Giovannucci et al., 2019). DeepLabCut benchmarking data consists of behavioral video capturing social interactions between two mice in their home cage. Data is provided courtesy of Robert C. Froemke and Ioana Carcea, as analyzed and presented in Carcea et al. (2019). Data processing consisted of analyzing these videos with a model that had previously been trained on other images from the same dataset. The same dataset was used to benchmark PMD and LocaNMF as a single analysis pipeline with two discrete parts. Input data consist of the dataset (“mSM30”), comprising widefield calcium imaging data videos, provided courtesy of Simon Musall and Anne Churchland, as used in Musall et al. (2019) and Saxena et al. (2020). The full dataset is available in a denoised format at http://repository.cshl.edu/id/eprint/38599/. Data processing on NeuroCAAS consisted of first processing the raw videos with PMD, then passing the resulting output to LocaNMF. For DLC and PMD+LocaNMF, the NeuroCAAS compute time was effectively constant across increasing total dataset size, as we assumed data was evenly batched into subsets of approximately equal size and each batch was analyzed in its own independent infrastructure stack (as in Figure 5A).

Further details on the datasets used can be found in Table S4.

We split the time taken to run analyses on NeuroCAAS into two separate quantities. First, we quantified the time taken to upload data from local machines to NeuroCAAS, denoted as NeuroCAAS (Upload) in Figure 7. This time depends upon the specifics of the internet connection that is being used. It is also a one time cost: once data is uploaded to NeuroCAAS, it can be reanalyzed many times without incurring this cost again. Upload times were measured from the same NeuroCAAS interface made available to the user. (This upload time was skipped in the quantification of local processing time.) Second, we automatically quantified the total time elapsed between job submission and job termination, when results have been delivered back to the end user in the NeuroCAAS interface (denoted as NeuroCAAS (Compute) in Figure 7) via AWS native tools (see supplemental information for details, and use of this data for Figure 3). Finally, we do not include time taken to download data back to a local machine in these quantifications because we found that this time was negligible across all analyses that we considered-at most $\sim$ 2 minutes, and in most cases on the order of a few seconds. Local timings were measured on automated portions of workflow in the same manner as NeuroCAAS (Compute).

We quantified the cost of running analysis on NeuroCAAS by enumerating costs of each of the AWS resources used in the course of a single analysis. Resources can be found in Table S7. We provide the raw quantification data and corresponding prices in Table S7 as well. To further reduce costs, we also offer the option to utilize AWS Spot Instances (dedicated duration); these are functionally identical to standard compute instances, but are provisioned for a pre-determined amount of time with the benefit of significantly reduced prices. We provide the estimated cost of running analyses with both of these options in Figure 7, with quantifications labeled “NeuroCAAS Save” corresponding to analyses run with dedicated duration spot instances, and those labeled “NeuroCAAS Std” corresponding to those run with standard instances. For more on Spot Instance price quantification, see supplemental information.

With simulated local infrastructures on AWS in hand, costs were calculated by pricing analogous computing resources as if the user had purchased them for a personal workstation, or as if they had been allocated to the user on an on-premises cluster (Table S8, https://calculator.aws/). In Figure 7, we assume that the local infrastructures considered are hosted on typical local laptop or desktop computing resources, supplemented with the resources necessary to run analyses as they were done on NeuroCAAS (additional storage, memory, GPU, etc), while maintaining approximate parity in processor power. We referred to Morey and Nambiar (2009) to convert pricetag costs of local machines to Equivalent Annual Costs, i.e. the effective cost per year if we assume our local machines will remain in service for a given number of years, as our implementation of a TCO calculation (as is often done in industry). Given a price tag cost x_{local}, an assumed lifetime n, an annuity rate r, and $c_s(n)$ defined as the estimated annual cost of local machine support given a lifetime n, we follow Mahvi and Zarfaty (2009) and Morey and Nambiar (2009) in calculating the Equivalent Annual Cost as:

$$EAC(x_{local}, n, r) = \frac{x_{local}}{\frac{1-(1+r)^{-n}}{r}} + c_s(n).$$

Here $c_s(n)$ is provided in the cited paper (Morey and Nambiar, 2009), estimated from representative data across many different industries. The denominator of the first term is an annuity factor. We consider two different values for n, which we label as "realistic" (2 years) and "optimistic" (4 years) in the text. In industry, 3–4 years is the generally accepted optimal lifespan for computers, after which support costs outweigh the value of maintaining an old machine (Business Intelligence, 2004, Mahvi and Zarfaty, 2009; Morey and Nambiar, 2009). Some have argued that with more modern hardware, the optimal refresh cycle has shortened to 2 years (J.Gold Associates LLC, 2014). By providing quantifications assuming two and four year refresh cycle, we consider the short and long end of this generally discussed optimal range.

Given a per-dataset NeuroCAAS cost $x_{\text{NeuroCAAS}}$, we further derive the Local Cost Crossover (LCC), the threshold weekly data analysis rate at which it becomes cost-effective to buy a local machine. The LCC is given by:

$$LCC(x_{local}, n, r, x_{\text{NeuroCAAS}}) = \frac{EAC(x_{local}, n, r)}{52 \times x_{\text{NeuroCAAS}}}.$$

Furthermore, given the per-dataset local analysis time, we can estimate the corresponding Local Utilization Crossover (LUC). The LUC considers the LCC in the context of the maximal achievable data analysis rate on local infrastructure as calculated in the previous section. If the time taken to analyze a dataset on a local machine is given by t_{local} (in seconds), the LUC is given by:

$$LUC(t_{local}, x_{local}, n, r, x_{\text{NeuroCAAS}}) = \frac{LCC(x_{local}, n, r, x_{\text{NeuroCAAS}}) \times t_{local} \times 100}{604800}.$$

Survey of analyses and platforms

We characterized data analysis infrastructure stacks as consisting of three hierarchical parts (Dependencies, System, Hardware), segmented consistently with infrastructure descriptions referenced elsewhere (Demchenko et al., 2013; Zhou et al., 2016). In several different subfields of neuroscience, we then selected 10 recent or prominent analysis techniques, and asked how they fulfilled each component of data analysis infrastructure to generate Figure 1D. We denoted a particular infrastructure component as supported if it is referenced in the relevant installation and usage guides as being provided in a reliable, automated manner (e.g., automatic package installation via pip), offering a conservative estimate of lack of infrastructure support. Survey details are provided in Tables S1 and S2. We addressed the question of how data analyses are installed and used with these surveys in the tradition of the open source usability literature. Surveys such as these are standard methodology in this field, which relies upon empirical data from studies of user's usage habits (Nichols et al., 2001; Zhao and Deek, 2005), developer sentiment (Terry et al., 2010), and observation of user-developer interactions via platforms like Github (Cheng and Guo, 2018).

To generate Figure 4, we first quantified the traffic and infrastructure experienced by individual analyses by examining their Github pages, and taking the maximum of the number of forks, stars, and watchers, as well as the listed hardware requirements of each analysis (numbers as of September 2020). We then overlaid several exemplar platforms based on the analyses that they supported, as well as restrictions based on the accessibility and scale requirements imposed by each (local hardware, limitation to one analysis at a time), taking care to include analyses that the platforms supported in practice.

ADDITIONAL RESOURCES

- Standard interface for users to work with analyses on NeuroCAAS: www.neurocaas.org.
- Documentation for developer workflow (and CLI usage): https://neurocaas.readthedocs.io/en/latest/develop/installation.html.