10. Supplemental Information

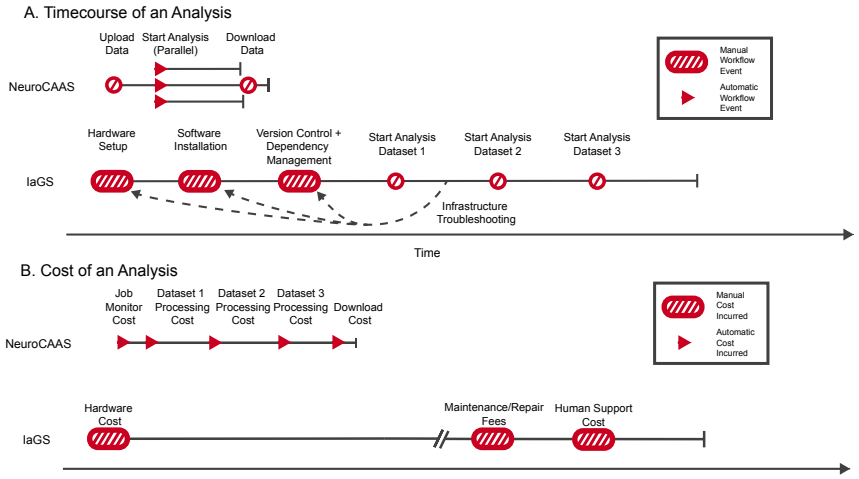

Figure S1: **Infrastructure-as-Grad-Student vs Infrastructure-as-Code.**, Related to Figure 2. A. Local processing via IaGS requires a number of time-consuming steps from the user (hardware setup, software installation and maintenance, etc.) before any analyses are run. Then typically analyses of large datasets are run serially (due to resource constraints), leading to longer processing times. Troubleshooting related to misconfiguration of infrastructure can cause massive delays. On NeuroCAAS, user interaction is only required at the beginning of the analysis (to upload the data), then NeuroCAAS processes the data using large-scale parallel compute resources, leading to faster overall processing times. User storage contains separated input and output areas where users can maintain datasets for re-analysis, or keep intermediate analysis results as convenient for subsequent jobs. From data in user storage, users can analyze multiple datasets in parallel. While errors may occur, users can delegate debugging to developers, as issues are guaranteed not to be due to infrastructure configuration. B. On NeuroCAAS, some costs are incurred with each analysis run: the user must upload the datasets (incurring a small job monitor cost), and then each dataset incurs some compute cost. The cost of using these analyses is directly proportional to analysis duration and the type of cloud resources used to construct the relevant infrastructure stack (see Table S7). For local processing, the bulk of the costs are paid upfront, in purchasing hardware; then additional labor costs are incurred for maintenance, support, and usage of limited local resources. If the per-dataset costs are low and the total number of datasets to be processed is limited then NeuroCAAS can lead to significantly smaller total costs than local processing §2.6.

```
{
    "PipelineName":"ncapexamplepipeline",
    "REGION":"region of service for users",
    "Lambda":{
        "CodeUri":"Codebase for \ncap Compute",
        "Handler":"Module for \ncap Compute",
        "Launch":"Whether or not to launch new pipelines. ",
        "LambdaConfig":{
            "AMI":"AMI id of the developer-configured instance",
            "INSTANCE_TYPE": "virtualized hardware instance id. ",
            "REGION": "us-east-1",
            "SECURITY_GROUPS":"network configuration",
            "IAM_ROLE":"permissions to launch new immutable analysis
                environments",
            "KEY_NAME":"permissions to access immutable analysis
                environments",
            "WORKING_DIRECTORY":"immutable analysis environment code",
            "COMMAND":"code to run to initiate processing",
            "SHUTDOWN_BEHAVIOR":"destroy immutable analysis environment
                after processing terminates",
            "CONFIG":"location of additional configuration parameters",
            "MISSING_CONFIG_ERROR":"We need a config file to analyze
                data.",
            "EXECUTION_TIMEOUT":"Additional parameters for \ncap Compute",
            "SSM_TIMEOUT":"Additional parameters for \ncap Compute",
            "LOGDIR":"Parameters for \ncap interface",
            "OUTDIR":"Parameters for \ncap interface",
            "INDIR":"Parameters for \ncap interface",
            "LAUNCH":"Launching new pipelines",
            "LOGFILE":"Logging location for diagnostic information",
            "DEPLOY_LIMIT":"Maximum number of concurrent instances to
                deploy",
            "MONITOR":"Enable or disable detailed monitoring"
        }
    },
    "UXData":{
    "Affiliates":[
        {
            "AffiliateName":"examplegroup1",
            "UserNames":["ian","shreya","taiga"],
            "UserInput":true,
            "ContactEmail":"The email we should notify regarding
                processing status."
        },
        {
            "AffiliateName":"examplegroup2",
            "UserNames":["liam","john"],
            "UserInput":true,
            "ContactEmail":"The email we should notify regarding
                processing status."
        }
    ]
    }
}
```


Figure S2: **NeuroCAAS blueprint template declaring all relevant resources.** Related to STAR methods. Immutable Analysis Environments can be defined from Variables in the Lambda.LambdaConfig field, the job manager protocol is defined in Lambda.CodeUri and Lambda.Handler. Users and permissions are defined in UXData.

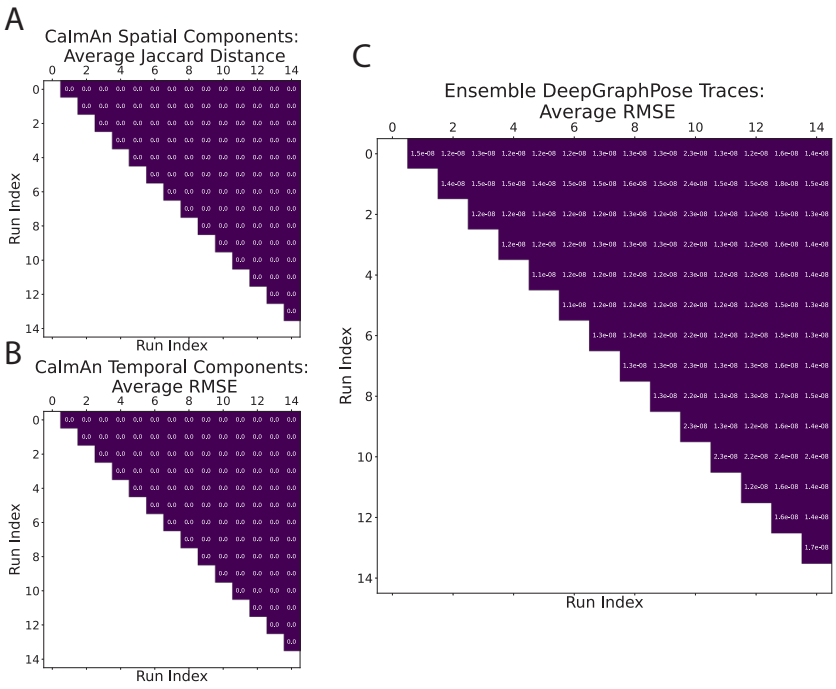

Figure S3: **Analyses on NeuroCAAS are reproducible.** Related to Table 1. Figure shows fifteen independent runs of two analysis algorithms- CaImAn on the left, and Ensemble DeepGraphPose on the right, run from different geographic locations and versions of the NeuroCAAS platform. The first five runs were performed by an author in the United States. The second five runs were performed by an independent researcher in India. The following four were performed by an independent researcher in Switzerland, and the last was run on a totally independent version of the NeuroCAAS platform. Each independent run uses the same data and configuration files, and assumes that all data is already uploaded to NeuroCAAS. Images show differences in the actual analysis outputs between runs. For CaImAn, we quantify the Jaccard Distance between detected spatial components (A), and the Average RMSE between detected temporal components (B). Analysis results are identical across all runs. For Ensemble DGP, we quantify the Average RMSE between body part traces produced by the same network (C). The average RMSE never exceeds order 1e-7. Given that these traces are given in units of pixels, these differences are negligible.

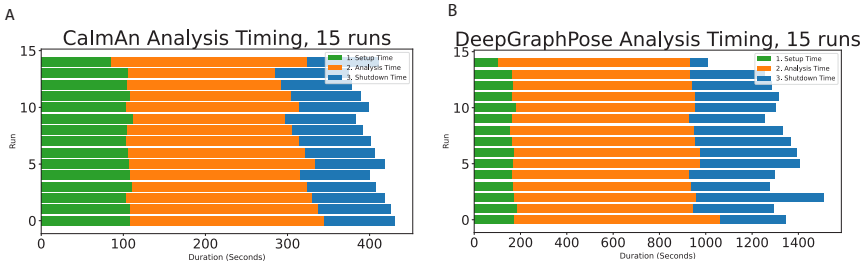

Figure S4: **Analyses on NeuroCAAS are reproducible.** Related to Table 1. Figure shows same fifteen independent runs of two analysis algorithms as in Figure S3. Images show the time taken to complete different parts of the analysis- in green is the time taken to set up analysis infrastructure after a request. In orange, we show the time between the successful setup of analysis infrastructure, and the production of the analysis's final output. In blue, we show the time required to take apart infrastructure after analysis is complete. Overall, across different runs differences in runs are small. These results emphasize the difference between our platform and others which offer public compute via cluster resources, where wait times in queues would affect such quantifications.

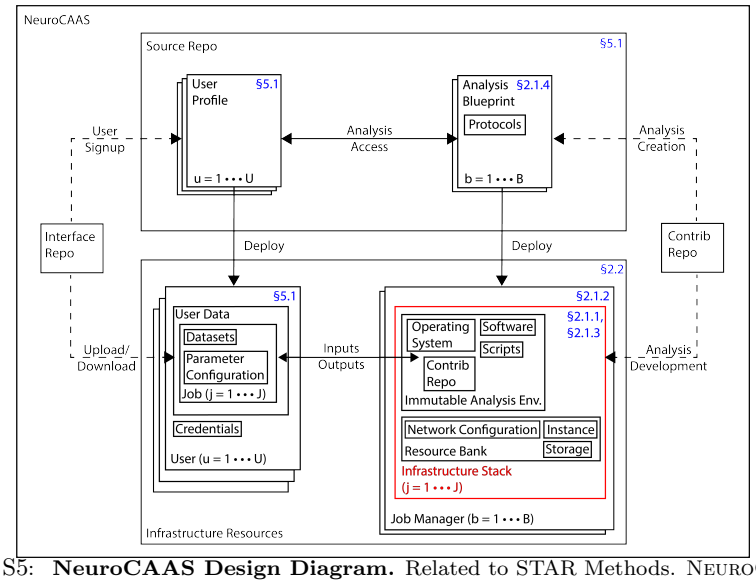

Figure S5: **NeuroCAAS Design Diagram.** Related to STAR Methods. NeuroCAAS is built with an Infrastructure-as-Code design, meaning that we first write a *source repo* (top) specifying all of the actual resources we will use to carry out data processing (bottom). The source repo (top) contains three main types of code: User Profiles, specifying relevant user data; Analysis Blueprints, describing individual analyses on NeuroCAAS, and Protocols, giving rules that describe NeuroCAAS job manager function. Each user and each analysis in NeuroCAAS has a dedicated code document, as specified by indices (u, b). All parts of the source repo can independently be *deployed*, automatically provisioning and configuring the infrastructure resources specified therein. Deployment comprehensively generates the resources necessary to run analyses on NeuroCAAS. Notably, most parts of infrastructure stacks (bottom right) are not persistent, but rather are instantiated every time users request an analysis job, specified as a combination of datasets and parameter configurations (bottom left). Job managers deploy one set of infrastructure for each dataset in a requested job, as specified by the index j. The contrib and interface repo assist in the deployment of resources from the source repo, and and the management of resulting resources. New users were registered by filling in a corresponding user profile with code from the interface repo, which was then deployed with the source repo to automatically generate storage space, dedicated login credentials, and permissions to use analyses for the user. The user profile is similar in format to the UXData segment of the blueprint as given in Figure S2, and can be found in the NeuroCAAS codebase online. Section numbers refer to relevant parts of the main text.

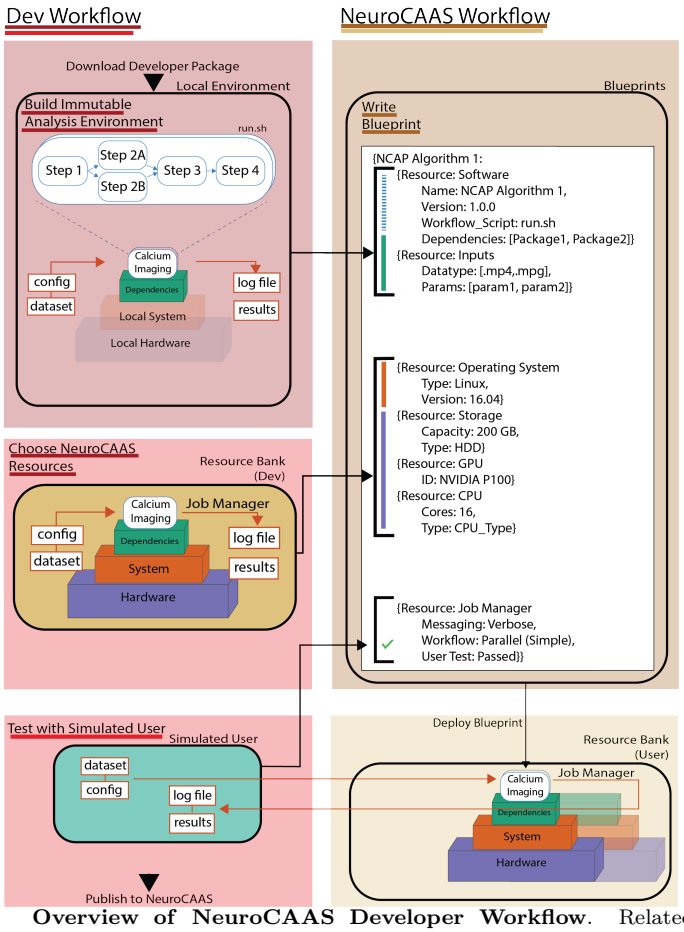

Figure S6: **Overview of NeuroCAAS Developer Workflow**. Related to STAR Methods. Left indicates the developer's experience; right indicates the work that NeuroCAAS performs. The developer begins by downloading the developer package `https://github.com/cunningham-lab/neurocaas_contrib` and building an immutable analysis environment script on their local machine. After determining workflow and optionally installing analysis software into an IAE, the developer locally tests that sample data and config files yield expected logs and results. Once satisfied, the developer updates a blueprint with IAE specifications. Next, developers configure system and hardware settings by setting up their IAE, complete with sample data and parameters, on hardware from the NeuroCAAS resource bank. Configuration and updates to an IAE are done identically to initial local build, allowing for IAEs custom-built for powerful cloud resources. Finally, developers simulate NeuroCAAS user accounts and trigger analyses with their blueprint to ensure that their analysis functions as intended end-to-end before publishing their blueprint for use.

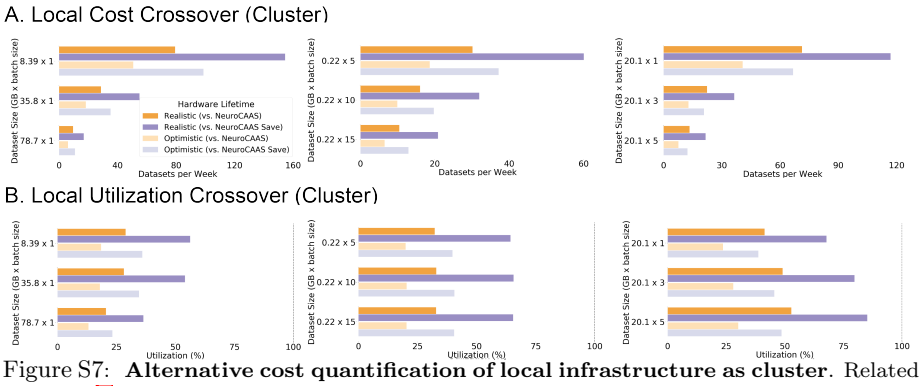

Figure S7: **Alternative cost quantification of local infrastructure as cluster**. Related to Figure 7. Because the instances offered on AWS are not wholly analogous to either personal hardware or cluster resources, we offer additional comparisons that span the range of prices. Cluster pricing was calculated with the AWS TCO calculator `https://calculator.aws/`. We calculated the cost of infrastructure as a subset of the TCO provided by AWS. In particular, we calculated x_{local} as the total server hardware cost (undiscounted) and acquisition cost of NAS storage, and the cost of a GPU, with additional yearly recurring costs $c_s(n)$ given by storage administration cost, server hardware maintenance cost, and IT Labor costs. We then calculated the LCC and LUC from these quantities as described in §2.6, 9.3. A) provides Local Cost Crossover Crossover for these resources priced as cluster compute resources, priced according to Amazon AWS's TCO calculator. B) provides the same for Local Utilization.

Calcium Imaging										
Algorithm Name	Publication	Software Version	Package Version	OS config	Batch Scripting	Other Processes	Storage	Memory	GPU	CPU
CaImAn	Giovannucci et al. 2017	✓	✓	X	✓	X	X	X	X	X
CNMF-E	Zhou et al. 2018	✓	✓	X	X	X	X	X	X	X
Suite2p	Pachitariu et al. 2017	✓	✓	X	✓	X	X	X	X	X
ABLE	Reynolds et al. 2017	✓	✓	X	✓	X	X	X	X	X
SCALPEL	Petersen et al. 2018	✓	✓	X	X	X	X	X	X	X
Min1PIPE	Lu et al. 2018	✓	✓	X	✓	X	X	X	X	X
SamuROI	Rueckl et al. 2017	✓	X	X	X	X	X	X	X	X
Romano	Romano et al. 2017	✓	✓	X	X	X	X	X	X	X
FISSA	Keemink et al. 2018	✓	✓	X	✓	X	X	X	X	X
OASIS	Friedrich et al. 2017	✓	✓	X	X	X	X	X	X	X
Percentage Supporting		100%	90%	0%	50%	0%	0%	0%	0%	0%

Table S1: **Infrastructure support for Calcium Imaging Algorithms.** Related to Figure 1. See STAR methods, 9.3 for details on evaluation.

Behavioral Quantification.										
Algorithm Name	Publication	Software Version	Package Version	OS config	Batch Scripting	Other Processes	Storage	Memory	GPU	CPU
DeepLabCut	Mathis et al. 2018	✓	✓	✓	X	X	X	X	X	X
DeepFly3D	Günel et al. 2019	✓	✓	X	X	X	X	X	X	X
JAABA	Kabra et al. 2012	✓	✓	X	X	X	X	X	X	X
Ctrax	Branson et al. 2009	✓	✓	X	✓	X	X	X	X	X
DeepPoseKit	Graving et al. 2019	✓	X	X	X	X	X	X	X	X
Ethovision	—.	✓	✓	X	X	X	X	X	X	X
APT	—.	✓	✓	✓	X	X	X	X	X	X
bonsai	Lopes et al. 2015	X	✓	X	X	X	X	X	X	X
Miceprofiler	de Chaumont et al. 2012	✓	✓	X	X	X	X	X	X	X
LEAP	Pereira et al. 2018	✓	✓	X	X	X	X	X	X	X
Percentage Supporting		90%	90%	20%	10%	0%	0%	0%	0%	0%

Table S2: **Infrastructure support for Behavioral Quantification Algorithms.** Related to Figure 1. See STAR methods, 9.3 for details on evaluation.

NeuroCAAS IAEs			
Analysis Name	Paper	Subfield	Description
DeepLabCut (dev, public, 2.X)	(Mathis et al., 2018)	Pose Tracking	Markerless pose estimation of user-defined features with deep learning for all animals, including humans.
DeepGraphPose	(Wu et al., 2020)	Pose Tracking	DGP is a semi supervised model which can run on top of other tracking algorithms, such as DLC.
Ensemble DGP	N/A	Pose Tracking	Ensemble DGP independently trains N different DeepGraphPose models that differ only in the sequence of minibatches that they see, and generates a consensus prediction from them.
DLC Tracking (Polleux)	N/A	Pose Tracking	(Custom) Markerless pose estimation with postprocessing to quantify freezing behavior.
DLC Tracking (Carcea)	N/A	Pose Tracking	(Custom) Markerless pose estimation with postprocessing to quantify social behavior.
Labeling GUI	N/A	Pose Tracking	Labeling GUI for a variety of pose tracking algorithms.
BehaveNet / Partitioned-subspace VAE	(Batty et al., 2019) (Whiteway et al., 2021)	Behavioral Video Analysis	Nonlinear embedding and Bayesian neural decoding of behavioral videos.
CaImAn (dev, public)	(Giovannucci et al., 2019)	Calcium Imaging	Integrated python toolbox for large scale Calcium Imaging data Analysis and behavioral analysis.
LocaNMF	(Saxena et al., 2020)	Widefield Imaging	Region-based Decomposition for Widefield Calcium Imaging Data.
PMD	(Buchanan et al., 2018)	Functional Imaging	Penalized Matrix Decomposition for Denoising and Compression of Functional Imaging Data.
WFCI Pipeline	Couto et al. (2021)	Widefield Imaging	Full data pipeline to work with widefield imaging data equipped with GUI
1-photon compression	N/A	Functional Imaging	GUI based compression of imaging data.
1-photon motion correction	N/A	Functional Imaging	GUI based motion correction of imaging data.
1-photon demixing	N/A	Functional Imaging	GUI based demixing of imaging data.
BarDensr	(Chen et al., 2020)	Spatial Transcriptomic Imaging	BarDensr (BARcode DEmixing through Non-negative Spatial Regression) for demixing spatial transcriptomic imaging data.
YASS	(Lee et al., 2017)	Spike Sorting	YASS is a spike sorting pipeline developed for high-firing rate, high-collision rate retinal recordings.
Latent Factor Analysis for Dynamical Systems: LFADS	(Sussillo et al., 2016)	Probabilistic Inference on Time Series	Deep learning method to infer latent dynamics from single-trial neural spiking data.
Kalman Filter/Smoother-Linear Dynamical System Inference.	(Minka, 1999)	Probabilistic Inference on Time Series	Time-invariant model for tracking a single object in a continuous state space.
Emergent Property Inference	(Bittner et al., 2019)	Likelihood Free Inference	A method for learning parameter distributions on models constrained to produce certain desirable phenomena in their output.

Table S3: **List of existing analyses currently implemented on NeuroCAAS through IAEs.** Related to Figure 2. Parentheses indicate different individual IAEs for a single analysis, corresponding to analysis versions or development stages.

Dataset Details							
Analysis	Format	Size (Small)	Dim (Small)	Size (Medium)	Dim (Medium)	Size (Large)	Dim (Large)
CaImAn	zipped tiff	8.39GB	$8000 \times 512 \times 512$[1]	35.84GB	$41000 \times 458 \times 477$[1]	78.70GB	$90000 \times 463 \times 472$[2]
DeepLabCut	mpeg	5×214.8MB	$5 \times 36000 \times 340 \times 420$[3]	10×214.8MB	$10 \times 36000 \times 340 \times 420$[3]	15×214.8MB	$15 \times 36000 \times 340 \times 420$[3]
PMD + LocaNMF	numpy array	1×20.1GB	$[500 \times 600 \times 1697, 1697 \times 8979]$[4]	3×20.1GB	$[500 \times 600 \times 1697, 1697 \times 8979]$; $[500 \times 600 \times 1652, 1652 \times 9000]$; $[500 \times 600 \times 2298, 2298 \times 8988]$[4]	5×20.1GB	$[500 \times 600 \times 1697, 1697 \times 8979]$; $[500 \times 600 \times 1652, 1652 \times 9000]$; $[500 \times 600 \times 2298, 2298 \times 8988]$; $[500 \times 600 \times 2304, 2304 \times 8992]$; $[500 \times 600 \times 1910, 1910 \times 8952]$[4]

1 [Time × X × Y] at 7 hz Giovannucci et al. (2019)
2 [Time × X × Y] at 30 hz Giovannucci et al. (2019)
3 [Batch × Time × X × Y] at 30 hz
4 [X × Y × Rank, Rank × Time] at 30 Hz

Table S4: **Details of the datasets used to benchmark performance.** Related to Figure 7. Sizes given for the three datasets tested for each pipeline shown. Dataset dimension labels are included in footnotes provided.

NeuroCAAS Benchmarked Analyses									
Analysis Name	Storage	Memory	GPU	CPU count	OS	Job Monitor	Resource Usage	Version Control	Packages
CaImAn	500 GB SSD	275 GB	N/A	64 vCPU[1]	Ubuntu 16.04 (Linux HVM)	stdout	CPU Utilization	Github	pip requirements file
DeepLabCut	200 GB SSD	65 GB	Nvidia Tesla K80	4 vCPU[2]	Ubuntu 16.04 (Linux HVM)	stdout	CPU Utilization	Github	conda environment file
PMD	75 GB SSD	275 GB	N/A	64 vCPU[1]	Ubuntu 18.04 (Linux HVM)	stdout	CPU Utilization	Github	Conda Package
LocaNMF	75 GB SSD	65 GB	Nvidia Tesla V100	8 vCPU[3]	Ubuntu 16.04 (Linux HVM)	stdout	CPU Utilization	Github	Conda Package

[1] Intel Xeon Platinum 8000 series (Skylake-SP)

[2] Intel Broadwell (AWS)

[3] Intel Xeon E5-2686v4

Table S5: **Infrastructure details for benchmarked algorithms.** Related to Figure 7. Job Monitor refers to the mechanisms used to track the status of ongoing jobs. Resource Usage refers to the hardware diagnostics tracked by NeuroCAAS. Version Control refers to the version control mechanisms used to maintain fidelity of core analysis code. Packages refers to the mechanisms used to handle analysis dependencies.

NeuroCAAS Local Simulation									
Analysis Name	Storage	Memory	GPU	CPU count	OS	Job Monitor	Resource Usage	Version Control	Packages
CaImAn	500 GB SSD	17 GB	N/A	4 vCPU[1]	Ubuntu 16.04 (Linux HVM)	stdout	None	github	Pip requirements file
DeepLabCut	200 GB SSD	65 GB	Nvidia Tesla K80	4 vCPU[2]	Ubuntu 16.04 (Linux HVM)	stdout	None	github	Conda Package
PMD	75 GB SSD	131 GB	N/A	16 vCPU[3]	Ubuntu 18.04 (Linux HVM)	stdout	None	Github	Conda Package
LocaNMF	75 GB SSD	65 GB	Nvidia Tesla K80	4 vCPU[2]	Ubuntu 16.04 (Linux HVM)	stdout	None	github	Conda Package

[1] AMD EPYC 7000 Series

[2] Intel Broadwell (AWS)

[3] Intel Xeon E5 Broadwell Processors

Table S6: **Details of infrastructure used to simulate local processing.** Related to Figure 7. The column labels mirror those in Table S5.

Pricing List		
Resource	Metrics	Rate
EC2 (Compute)	Time	Hardware Dependent, Fluctuates
Lambda (Workflow)[1]	Data Size × Time	$1.66667×1e−5 per GB-second
S3 (Data Transfer Out)[2]	Data Size	$0.09 per GB

[1] AWS Lambda is also priced for number of requests, but this is a negligible cost for a single analysis run.

[2] Data Transfer is only priced out of Amazon Web Services, i.e. in returning results to the end user.

Table S7: **Pricing details for implemented algorithms.** Related to Figure 7. The virtualized hardware underlying a hardware instance can be provisioned at several different prices. We used AWS EC2 Spot Instance pricing to reduce costs, having known beforehands how long the analyses would take. At the moment, we depict prices based on spot instance availability in September 2019. Empirically, we observe that spot instance price fluctuations give standard deviations on the order of cents over a period of months (see source repo for experiments). The duration of NeuroCAAS Compute and Local analysis time was recorded automatically with cloud native resource monitoring tools (AWS Lambda, AWS Cloudwatch Events, and AWS S3). These tools automatically recorded the creation and destruction of instances, and recorded the relevant timestamps at millisecond resolution. These monitoring tools were also managed via NeuroCAAS blueprints, and their design can be found in the same blueprint codebase. The same tools were used to calculate the usage data shwon in Figure 3. We do not disclose user data at an individual level, but developers can generate the same figures for users of their own analysis by using the NeuroCAAS contrib repo (specific instructions are given in the source repo's README file).

Cost (Local)						
Algorithm Name	vCPU count	GPU	Memory	Storage Capacity	Workstation Price, US Dollars (Estimated Price Tag Cost)	Cluster Price, US Dollars (Estimated Price Tag Cost from Amazon TCO Calculator)
CaImAn	4	N/A	16 GiB	500 GB	1618[2]	1499+1000
DeepLabCut	4	Tesla K80	61GiB	200 GB	3120[3]	1701+400+1555[5]
PMD + LocaNMF [1]	16	Tesla K80	122 GiB	150 GB	5436[4]	10836+300+1555[5]

[1] Hardware cost for a local instance that can account for processing done on all analyses.

[2] CaImAn Hardware Price

[3] DeepLabCut Hardware Price

[4] PMD+LocaNMF cost

[5] Cluster Price

Table S8: **Instance and hardware cost details for local cost comparisons.** Related to Figure 7. Estimated Price tag prices as of May 3rd, 2020. Price tag estimation of workstation style hardware was based on market prices chosen to reflect the infrastructure implementation as given in Table S6, in particular, CPU make. Estimation of cluster style hardware cost was based on the AWS TCO calculator (`https://awstcocalculator.com`), as of January 25th, 2020, incorporating the total server hardware cost (undiscounted) and acquisition cost of SAN storage.