A modular architecture for organizing, processing and sharing neurophysiology data

The International Brain Laboratory, Niccolò Bonacchi[1], Gaelle A. Chapuis[2,3], Anne K. Churchland[4], Eric E. J. DeWitt[1], Mayo Faulkner[2], Kenneth D. Harris[2], Julia M. Huntenburg[5], Max Hunter[2], Inês C. Laranjeira[1], Cyrille Rossant[2], Maho Sasaki[6], Michael M. Schartner[1], Shan Shen[6], Nicholas A. Steinmetz[7], Edgar Y. Walker[6], Steven J. West[8], Olivier Winter[1], Miles Wells[2]

[1]Champalimaud Center for the Unknown, Av. Brasília, 1400-038 Lisboa, Portugal

[2]Institute of Neurology, University College London, London WC1N 3BG, UK

[3]Department of Basic Neuroscience, University of Geneva, Geneva, Switzerland

[4]Department of Neurobiology, University of California, Los Angeles, CA 90095, USA

[5]Max Planck Institute for Biological Cybernetics, 72076 Tübingen, Germany

[6]DataJoint, Houston, TX 77027, USA

[7]Department of Biological Structure, University of Washington, Seattle, WA 98195, USA

[8]Sainsbury-Wellcome Centre, University College London, London WC1N 3BG, UK

Correspondence: info+data@internationalbrainlab.org, kenneth.harris@ucl.ac.uk

Abstract

We describe an architecture for organizing, integrating, and sharing neurophysiology data in single labs or collaborations. It comprises a database linking data files to metadata and electronic lab notes; a module collecting data from multiple labs into one location; a protocol for searching and sharing data; and a module for automatic analyses that populates a website. These modules can be used together or individually, by single labs or worldwide collaborations.

Main text

Improving technology allows neurophysiologists to record ever larger datasets. The need for technologies to organize and share this data is growing as scientists begin to assemble into large, international teams. The International Brain Laboratory (IBL) is a collaboration studying the computations supporting decision-making in the mouse[1]. We have developed modular data-management tools that enable individual labs and collaborations to:

- Manage experimental subject colonies and track subject- and experiment-level metadata
- Integrate data from multiple labs in a central store for sharing inside or outside the collaboration
- Access shared data through a programmatic interface
- Process incoming data through pipelines that automatically populate a website

Current neurophysiological datasets comprise multiple recordings from multiple subjects, recorded using diverse devices. These data must be preprocessed, time-aligned, and integrated with data such as locations of recording electrodes before they can be used to draw scientific conclusions[2–8]. Distributed collaborations pose distinct challenges: while public data release must wait for careful quality control, scientists within the collaboration require immediate access to specific data. This store must be searchable and allow downloading and also revision of individual items, because preprocessing and quality control methods are still evolving[9–11].

We addressed these problems with an architecture consisting of four modules (Figure 1). The first module is a Web interface for colony management and electronic lab notebook, that links files arising from each experiment to relevant metadata. The second module integrates data from multiple labs into a central database and bulk data store, providing immediate access while allowing updates of individual items. The third automatically runs analyses on newly arrived data, providing results via a Web interface. The fourth allows standardization, access and sharing of the data. Full documentation can be found at https://docs.internationalbrainlab.org/ and through links at https://www.internationalbrainlab.com/tools.

To manage data within each lab, we developed Alyx, a relational database that links colony management, metadata, and lab notes to experimental data files. A web GUI allows users to enter metadata as it arrives (such as birth, weaning, genotyping, surgeries or experiments), and a REST API allows experiment control software to automatically enter metadata with a one-line command. Bulk data files are stored on a lab server and linked to experiment and subject metadata in the database. This tool can be used by single labs as well as collaborations: it was developed in one member lab prior to IBL's founding, and is now used by several labs worldwide for non-IBL work. An Alyx user guide can be found here, or linked via our main documentation page.

Integrating data between labs raises challenges of size and complexity. Large-scale electrophysiology produces hundreds of gigabytes per experiment, for which we have designed a 3-fold lossless compression algorithm (Supplementary Note 1). A single IBL experiment generates over 150 raw and processed data files. We have devised conventions for organizing and naming these files, termed the "Open Neurophysiology Environment" (ONE; Supplementary Note 2; https://int-brain-lab.github.io/ONE/), which formalizes how to encode cross-references between files, time synchronization, and versioning, and allows local and remote access via an API. ONE provides a way to standardize and share data from individual labs, by specifying standard filenames for common data types (Supplementary Note 3) and defining conventions for naming lab-specific data files. Files from multiple labs are integrated by uploading nightly from lab servers to a central server using Globus Online[12], coordinated by a central Alyx database that also stores metadata from all labs.

Neurophysiology data requires preprocessing, such as spike sorting and video analysis. We developed a task management system that uses computers in member labs as a processing pool. Computers query the Alyx database for a list of outstanding preprocessing tasks, determined by a dependency graph. Because Alyx is accessed through http, this works despite different universities' diverse firewall policies, and allows monitoring, logging, and restarting all preprocessing tasks. Higher-level analyses are automatically run on newly preprocessed data using DataJoint[14], which runs automated analyses and places the results on a website, including summaries of behavioral performance, allowing scientists to monitor training progress, and basic

analyses of spike trains. While manual curation of the full dataset will be required before public release, an illustrative curated subset of these data are available on a public website (https://data.internationalbrainlab.org).

To access data, an API allows users to search experiments and load data from the ONE files directly into Python (Supplementary Note 3). This API allows both collaborations and individual labs to share data using the same standard. A large collaboration such as IBL can host files on a server such as AWS, and run an Alyx server which allows users to rapidly search and selectively download the data. Individual labs can release data compatible with the same API by "uploading and forgetting" a zip of ONE files for users to download in toto (instructions here). Users can also access data via Neurodata Without Borders (NWB)[13,14] using software that translates from the ONE standard (https://github.com/catalystneuro/IBL-to-nwb; Supplementary Table 1), or through DataJoint[15]. A comparison of these and other sharing systems is in Supplementary Note 4. The analyses in a recently published paper[1] were made using this system, and an additional example is provided in Supplementary Note 5.

The IBL architecture was designed for our large-scale collaboration, but its modular design allows components to be used by individual labs and smaller-scale collaborations. The Alyx system provides easy-to-use colony management and electronic lab notebook features for labs or collaborations, linking experimental files to this metadata. The ONE conventions allow data to be organized within a lab and shared externally, using standards that scale to large collaborations. Larger collaborations can also benefit from other features such as the DataJoint architecture to perform automated analyses for web display. We hope that these tools, and additional software we have provided (Supplementary Table 1), will help pave the way forward to an era in which data from neurophysiology labs is integrated and shared on a routine basis.

Example use case: evaluating training time

To demonstrate how this system can manage data and metadata, integrate them across labs, and analyze the results, we evaluated the importance of multiple variables for predicting the time required for mice to complete behavioral training.

Mice were on a visual discrimination task using the standard IBL training pipeline[1]. Training was considered complete when performance met criteria for the fraction of correct responses, number of completed trials, and fitted psychometric parameters, for 3 consecutive sessions. Behavior upon reaching this criterion was similar across mice, but the training time required for mice to meet these criteria was variable, ranging from 5 to 57 training sessions (Fig. 2A). We used the data architecture described above to investigate which factors might predict this variability. Because comprehensive data and metadata from all laboratories were integrated in a centralized and standardized manner, we could quickly perform these analyses.

We investigated whether training time could be predicted from several classes of variables. The first class was subject features: the sex of the animal, the age, weight and weight loss (relative to pre-water-restriction weight) upon training start. The second was rig ambient measures:

temperature, relative humidity, and air pressure, averaged across all training sessions. Third, some institute-specific experimental conditions such as the type of light cycle mice were housed in, the protein content of the homecage food, and the weekend water regime in place (water restriction versus 2% free homecage citric acid water[16]). Fourth, metrics assessed from early training sessions including: task performance; median reaction time; total number of trials on the first training session; the changes in those values over the first 5 training sessions; the total sum of trials performed over the first 5 training sessions; the variance in the sign of the daily performance change across the first 5 training sessions; the number of wheel movements per second and the average wheel displacement bias (averaged across the first 5 training sessions).

A random forest classifier accurately predicted time to reach the performance criterion for each mouse from this feature set (Fig. 2A). Time to criterion was grouped into quartiles and classification accuracy was evaluated by 10-fold cross-validation, producing a confusion matrix comparing the predicted and actual quartile for each mouse (Fig. 2B), summarized by an F1 score (Fig. 2C). When trained with all available features, the classifier predicted the true quartile more often than any other (Fig. 2B), with accuracy around two times higher than when trained after randomly shuffling quartile labels (Fig. 2C).

To investigate the importance of each feature, we performed a permutation test on each of the features. The importance of each feature was assessed by the decrease in the classifier's accuracy after randomly shuffling that feature's values across mice. This revealed that one predictor variable was more important than all others: the task performance change across the first 5 training sessions (Fig. 2D), i.e. the percent correct achieved on session 5 minus the percent correct achieved on session 1. Site-specific features that are hard to standardize across locations, such as food protein content and humidity, were not important to the classifier's accuracy. The only predictive feature not related to task performance in early days was age.

Given the importance of the 5-day performance change feature compared to the remaining ones, we further evaluated the accuracy of a classifier trained only with this one feature (Fig. 2C). Prediction using only this feature was nearly as accurate as the full classifier, although including other predictor variables resulted in a 14% increase in accuracy.

This large-scale analysis was made possible by the ease and speed of accessing large amounts of behavioral data saved in a standard manner. The obtained results showed that tracking changes in performance during the first few training days was enough to predict training time above chance level, with even better accuracy achieved when also considering other behavioral metrics. The ability to predict final training time after only 5 training sessions could allow automated decisions about when to drop a subject from the training pipeline.

Methods

The experimental methods used to collect the data analyzed in this paper are described in Ref.

Acknowledgements

This work was supported by the Wellcome Trust (209558 to IBL, 216324 to IBL) and Simons Foundation (to IBL).

Author contributions

Niccolò Bonacchi [Conceptualization] (supporting). [Data curation] Data, metadata, and pipeline (equal). [Funding acquisition] (supporting). [Project administration] Meeting coordination (supporting), attendance (equal). [Resources] Computing and data storage (equal). [Software, Validation] Pipeline, core, quality control (equal), database and analysis libraries (supporting).. [Writing – original draft] (equal). [Writing – review & editing] (equal).

Gaelle Chapuis [Data Curation] Helped with writing user guides detailing how to enter metadata in Alyx (supporting). [Software, Validation] Acted as naive user tester for ONE and DJ (supporting). [Project administration] Gathered and reported users' requirements (supporting).

Anne Churchland Contributed to project administration, funding acquisition, writing and revising.

Eric E. J. DeWitt [Investigation] (supporting) Contributed to analyses for example use of data [Writing] (supporting) Figures and draft text for example.

Mayo Faulkner. [Software] Contributed to implementation of backend data infrastructure and analysis libraries. [Data Curation] Contributed to curating datasets and assuring quality assurance.

Kenneth D. Harris. Contributed to design of overall data architecture, and to Alyx and ONE systems. Contributed to project administration, funding acquisition, writing and revising.

Julia Huntenburg. [Software] Contributed to implementation of backend data infrastructure, analysis libraries and continuous integration. [Data Curation] Contributed to dataset curation and quality assurance.

Michael Schartner. [Conceptualisation] Contributed to the development of dataset types related to video and their quality control metrics.

Max Hunter. Contributed to the design and development of Alyx.

Inês Laranjeira. [Investigation] Performed analyses for the example use case of the data architecture (lead). [Writing - original draft] (supporting)

Cyrille Rossant. Contributed to design and implementation of overall data architecture, and to Alyx and ONE systems.

Maho Sasaki Contributed to the design and implementation of the IBL Data Portal website.

Shan Shen Contributed to the design and implementation of the DataJoint pipeline and the IBL JupyterHub

Nicholas A. Steinmetz contributed to the design, testing, and development of Alyx, of dataset types, and of software tools for working with them.

Edgar Y. Walker Contributed to the design and implementation of the DataJoint pipeline and the IBL Data Portal.

Steven J. West contributed to the design of data structures for histological alignment.

Olivier Winter [Software] Implemented the backend data infrastructure. [Validation/Methodology] Designed full loop integration tests to allow maintenance of the codebase. [Data Curation] Fixed and updated erroneous datasets.

Miles Wells. [Software] Contributed to the design, testing and implementation of Alyx, its dataset types and of software tools that work with them. [Software, Validation] Contributed to the design and implementation of continuous integration and quality assurance systems. [Writing] Contributed to writing, reviewing and editing the text and Supplementary Notes.

Competing Interests

The authors declare the following competing interests: E.Y.W. holds equity ownership in Vathes LLC, which provides development and consulting for the framework (DataJoint) described in this work. E.Y.W., M.S., and S.S were employees of Vathes LLC at the time the work in this paper was done. The remaining authors declare no competing interests.

Figure legends

Figure 1*. IBL data architecture. The Alyx database links colony management and electronic lab notebook metadata to experimental data files on a lab data server. Data from multiple labs are integrated on a central server, and distributed job management coordinates pre-processing on lab servers. Data are accessed via the Open Neurophysiology Environment (ONE) protocol, with adaptors for Neurodata Without Borders (NWB)[12,13] and DataJoint [14], which also performs pipelined analyses for automatic display on a website.*

Figure 2 - *Predicting time taken to complete training from diverse data and metadata.* ***A.*** *Histogram of the number of training sessions taken to reach the IBL 'trained' criterion (N=116 mice). Vertical dashed lines represent the split of the data in quartiles.* ***B.*** *Cross-validated confusion matrix of a random forest classifier, trained to predict training time quantile from multiple behavioral features. Rows represent the true quartile and columns represent the predicted quartile; results were normalized over the number of mice of the corresponding true quartile (row).* ***C.*** *Prediction accuracy for a classifier that uses all features (Full classifier), and a classifier that uses only task performance change across the first 5 training sessions (Task performance change classifier). Horizontal lines show classifier performance; boxplots show distribution of performance scores over random shuffles of the training-time labels (N=100 shuffles).* ***D.*** *Importance of each feature in predicting training time. Box plots show the distribution of importance scores obtained across multiple permutations (N=10 permutations). In all boxplots, the box shows median and interquartile range, whiskers show range, points show individual observations.*

References

1. The International Brain Laboratory *et al.* Standardized and reproducible measurement of decision-making in mice. *eLife* **10**, e63711 (2021).

2. Pachitariu, M. *et al.* Suite2p: beyond 10,000 neurons with standard two-photon microscopy. *BioRxiv* **061507**, (2016).
3. Mathis, A. *et al.* DeepLabCut: markerless pose estimation of user-defined body parts with deep learning. *Nat. Neurosci.* **21**, 1281–1289 (2018).
4. Giovannucci, A. *et al.* CaImAn an open source tool for scalable calcium imaging data analysis. *eLife* **8**, e38173 (2019).
5. Vogelstein, J. T. *et al.* Fast nonnegative deconvolution for spike train inference from population calcium imaging. *J Neurophysiol* **104**, 3691–704 (2010).
6. Pachitariu, M., Steinmetz, N. A., Kadir, S. N., Carandini, M. & Harris, K. D. Fast and accurate spike sorting of high-channel count probes with KiloSort. in *Advances in Neural Information Processing Systems 29* (eds. Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I. & Garnett, R.) 4448–4456 (Curran Associates, Inc., 2016).
7. Wiltschko, A. B. *et al.* Revealing the structure of pharmacobehavioral space through motion sequencing. *Nat. Neurosci.* **23**, 1433–1443 (2020).
8. Vogelstein, J. T. *et al.* Discovery of Brainwide Neural-Behavioral Maps via Multiscale Unsupervised Structure Learning. *Science* **344**, 386–392 (2014).
9. Siegle, J. H. *et al.* Survey of spiking in the mouse visual system reveals functional hierarchy. *Nature* **592**, 86–92 (2021).
10. Hill, D. N., Mehta, S. B. & Kleinfeld, D. Quality metrics to accompany spike sorting of extracellular signals. *J Neurosci* **31**, 8699–705 (2011).
11. Harris, K. D., Quiroga, R. Q., Freeman, J. & Smith, S. L. Improving data quality in neuronal population recordings. *Nat. Neurosci.* **19**, 1165–1174 (2016).
12. Foster, I. Globus Online: Accelerating and Democratizing Science through Cloud-Based Services. *IEEE Internet Comput.* **15**, 70–73 (2011).
13. Teeters, J. L. *et al.* Neurodata Without Borders: Creating a Common Data Format for Neurophysiology. *Neuron* **88**, 629–634 (2015).

14. Rübel, O. *et al.* The Neurodata Without Borders ecosystem for neurophysiological data science. *eLife* **11**, e78362 (2022).

15. Yatsenko, D. *et al.* DataJoint: managing big scientific data using MATLAB or Python. *bioRxiv* 031658 (2015) doi:10.1101/031658.

16. Urai, A. E. *et al.* Citric Acid Water as an Alternative to Water Restriction for High-Yield Mouse Behavior. *eneuro* **8**, ENEURO.0230-20.2020 (2021).

Methods

The experimental methods used to collect the data analyzed in this paper are described in Ref. [1].

For the analysis described in this paper, we accessed the behavioral data using the public DataJoint protocol. Mice selected for the analysis consisted of all mice trained according to the standard IBL training pipeline, up until March 23 2020. Mice were excluded from the analyses if they were dropped from the pipeline before reaching the end of training. Training was considered complete when performance met criteria for the fraction of correct responses, number of completed trials, and fitted psychometric parameters, for 3 consecutive sessions[1].

A Random Forest classifier was used to assess whether training time could be predicted from several classes of variables: subject features, rig ambient measures, institute-specific experimental conditions, and performance metrics from early training sessions. For that, data were processed and organized as a design matrix with shape #mice x #variables. For each mouse, we included the following variables: (1) sex; (2) age at the start of training; (3) weight at the start of training; (4) weight loss at the start of training, calculated as the weight fraction relative to the pre-water-restriction weight; (5) whether the mouse was housed on an inverted or non-inverted light cycle scheme; (6) the percentage of protein content of the homecage food; (7) weekend water regime in place: whether mice were on a traditional water restriction regime or on had free access to 2% free homecage citric acid water[16]; (8) the training rig temperature, averaged across the first 5 training sessions; (9) the training rig relative humidity, averaged across the first 5 training sessions; (10) the training rig air pressure, averaged across the first 5 training sessions; (11) the fraction of correct responses on the first training session; (12) median reaction time on the first training session; (13) total number of trials on the first training session; (14) difference in fraction of correct responses between first and fifth training sessions; (15) difference in the median reaction time between the first and fifth training sessions; (16) difference in the total number of trials between the first and fifth training sessions; (17) total number of trials performed over the first 5 training sessions; (18) the variance in the sign of the daily performance change across the first 5 training sessions (daily performance change was computed as the difference in the fraction of correct responses across consecutive sessions); (19) the amount of wheel movement per second averaged across the first 5 training sessions; (20) the wheel displacement bias averaged across the first 5 training sessions (wheeldisplacement bias was calculated as the amount of wheel displacement divided by the

total amount of wheel movement). Missing data which prevented the calculation of any of the above metrics led to the exclusion of the corresponding mouse from the analyses. The predicted variable was the training time quartile of the mouse. Training time was calculated as the number of training sessions until training completion. The quartiles of the distribution were calculated after exclusion of mice with missing data.

To assess whether training time could be predicted from the listed variables, a Random Forest Classifier was trained on the data, using 10-fold cross-validation. For that, *scikit-learn* functions RanfomForestClassifier and KFold were used. Prediction accuracy of the classifier was computed using the f1-score function. The F1-score reaches 1 for the highest accuracy value and 0 for the worst. It is calculated according to the following formula:

$$F_1 = \frac{2 * true\ positives}{2 * true\ positives + false\ positives + false\ negatives}$$

Classifier performance was compared with that of a classifier trained on a control dataset in which quartile labels were randomly shuffled (N=100 shuffles).
To investigate the importance of each feature to the classifier's performance, we performed a permutation test on each of the features. The importance of each feature was assessed by the decrease in the classifier's accuracy (f1-score) after randomly shuffling that feature's values across mice (N=10 repetitions).

Finally, we further evaluated the accuracy of a classifier trained only on the most important feature, as concluded from the permutation test: the difference in fraction of correct responses between first and fifth training sessions.

The code used to make the figure is available at https://github.com/int-brain-lab/paper-data-architecture.

Reporting Summary Further information on research design is available in the Nature Research Reporting Summary linked to this article.

Data availability

All IBL data is available online using the access protocols described in this manuscript. For further information see https://www.internationalbrainlab.com/data. The specific data used to create Supplementary Figure 1 can be accessed by the code that created this figure, available at https://github.com/int-brain-lab/paper-data-architecture.

Code availability

All code described in this manuscript is freely available and is listed in Supplementary Table 1 along with links to their respective repositories. The behavior data were collected using Bonsai and pyBpod, available at https://github.com/int-brain-lab/iblrig. Meta data were stored in a custom database available at https://github.com/cortex-lab/alyx. The data were processed using the custom data pipelines ibllib (https://github.com/int-brain-lab/iblrig) and DataJoint

(https://datajoint.io/). The data were accessed using ONE (https://github.com/int-brain-lab/ONE) and DataJoint (https://github.com/int-brain-lab/IBL-pipeline).

Full List of Consortium Authors

Luigi Acerbi[3], Valeria Aguillon-Rodriguez[9], Mandana Ahmadi[10], Jaweria Amjad[10], Dora Angelaki[11], Jaime Arlandis[1], Zoe C. Ashwood[12], Kush Banga[2], Hailey Barrell[7], Hannah M. Bayer[13], Julius Benson[11], Brandon Benson[14], Jai Bhagat[15], Dan Birman[7], Niccolò Bonacchi[1], Kcenia Bougrova[1], Julien Boussard[13], Sebastian A. Bruijns[5], Matteo Carandini[15], Joana Catarino[1], Fanny Cazettes[1], Gaelle A. Chapuis[2,3], Anne K. Churchland[4], Yang Dan[16], Felicia Davatolagh[4], Peter Dayan[5], Sophie Denève[17], Eric E.J. DeWitt[1], Ling Liang Dong[18], Tatiana Engel[9], Michele Fabbri[13], Mayo Faulkner[2], Ila Fiete[18], Charles Findling[3], Laura Freitas-Silva[1], Surya Ganguli[14], Berk Gercek[3], Naureen Ghani[8], Ivan Gordeliy[17], Laura M. Haetzel[12], Kenneth D. Harris[2], Michael Hausser[19], Naoki Hiratani[10], Sonja Hofer[8], Fei Hu[16], Felix Huber[3], Julia M. Huntenburg[5], Cole Hurwitz[13], Anup Khanal[4], Christopher S. Krasniak[9, 20], Sanjukta Krishnagopal[10], Michael Krumin[2], Christopher Langdon[9], Inês C. Laranjeira[1], Peter Latham[10], Petrina Lau[19], Hyun Lee[13], Ari Liu[18], Zachary F. Mainen[1], Hernando Martinez Vergara[8], Conor Mcgrory[9], Brenna McMannon[12], Guido T. Meijer[1], Maxwell Melin[4], Leenoy Meshulam[7], Nathaniel J. Miska[8], Catalin Mitelut[13], Zeinab Mohammadi[12], Thomas Mrsic-Flogel[8], Masayoshi Murakami[1, 24], Jean-Paul Noel[11], Kai Nylund[7], Alex Pan Vazquez[12], Liam Paninski[13], Alberto Pezzotta[10], Samuel Picard[2], Jonathan W. Pillow[12], Alexandre Pouget[3], Cyrille Rossant[2], Noam Roth[7], Nicholas A. Roy[12], Kamron Saniee[13], Rylan Schaeffer[18,22], Michael M. Schartner[1], Yanliang Shi[9], Karolina Z. Socha[15], Cristian Soitu[9], Nicholas A. Steinmetz[7], Karel Svoboda[21], Marsa Taheri[4], Charline Tessereau[5], Anne E. Urai[9,23], Erdem Varol[13], Miles J. Wells[2], Steven J. West[8], Matthew R. Whiteway[13], Charles Windolf[13], Olivier Winter[1], Ilana Witten[12], Lauren E. Wool[2], Anthony M. Zador[9]

[9] Cold Spring Harbor Laboratory, Cold Spring Harbor, NY, USA
[10] Gatsby Computational Neuroscience Unity, University College London, London, United Kingdom
[11] Center for Neural Science, New York University, New York, NY, USA
[12] Princeton Neuroscience Institute, Princeton University, Princeton, NJ, USA
[13] Zuckerman Institute, Columbia University, New York, NY, USA
[14] Department of Applied Physics, Stanford University, Stanford, CA, USA
[15] Institute of Opthalmology, University College London, London, United Kingdom
[16] Department of Molecular and Cell Biology, University of California, Berkeley, CA, USA
[17] Département D'études Cognitives, École Normale Supérieure, Paris, France
[18] Department of Brain and Cognitive Sciences, Massachusetts Institute of Technology, Cambridge, MA, USA
[19] Wolfson Institute of Biomedical Research, University College London, London, United Kingdom
[20] Watson School of Biological Science, Cold Spring Harbor, NY, USA
[21] The Allen Institute for Neural Dynamics, Seattle, Washington, USA
[22] current address: Department of Computer Science, Stanford University, Stanford, CA, USA
[23] current address: Cognitive Psychology Unit, Institute of Psychology and Leiden Institute for Brain and Cognition, Leiden University, Leiden, Netherlands
[24] current address: Department of Physiology, University of Yamanashi, Kofu, Yamanashi, Japan

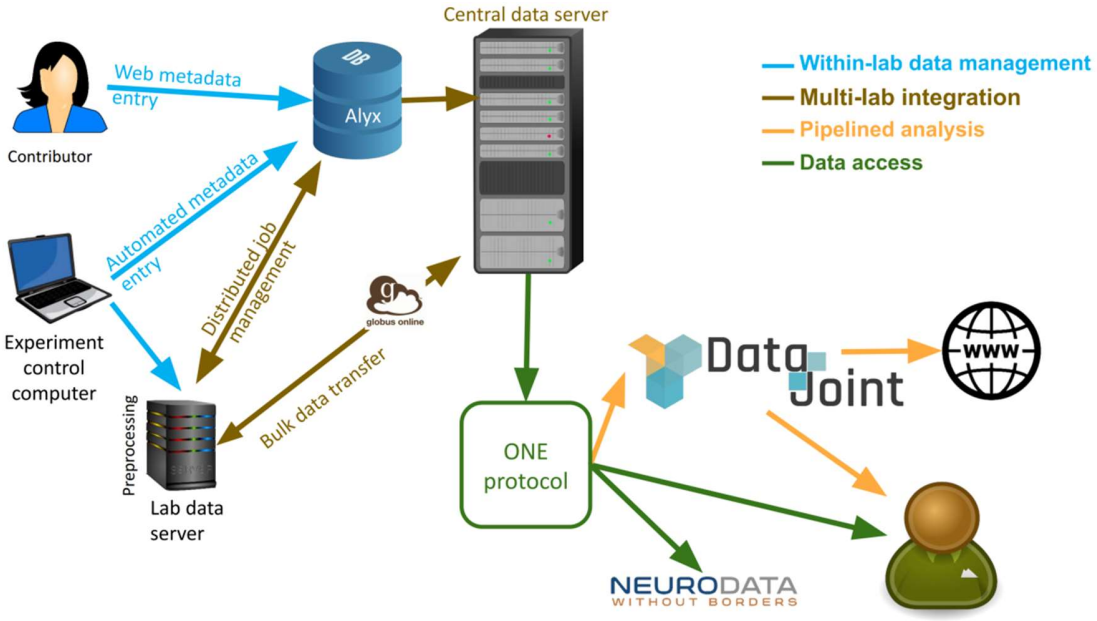

Figure 1. *IBL data architecture. The Alyx database links colony management and electronic lab notebook metadata to experimental data files on a lab data server. Data from multiple labs are integrated on a central server, and distributed job management coordinates pre-processing on lab servers. Data are accessed via the Open Neurophysiology Environment (ONE) protocol, with adaptors for Neurodata Without Borders (NWB)[12,13] and DataJoint [14], which also performs pipelined analyses for automatic display on a website.*

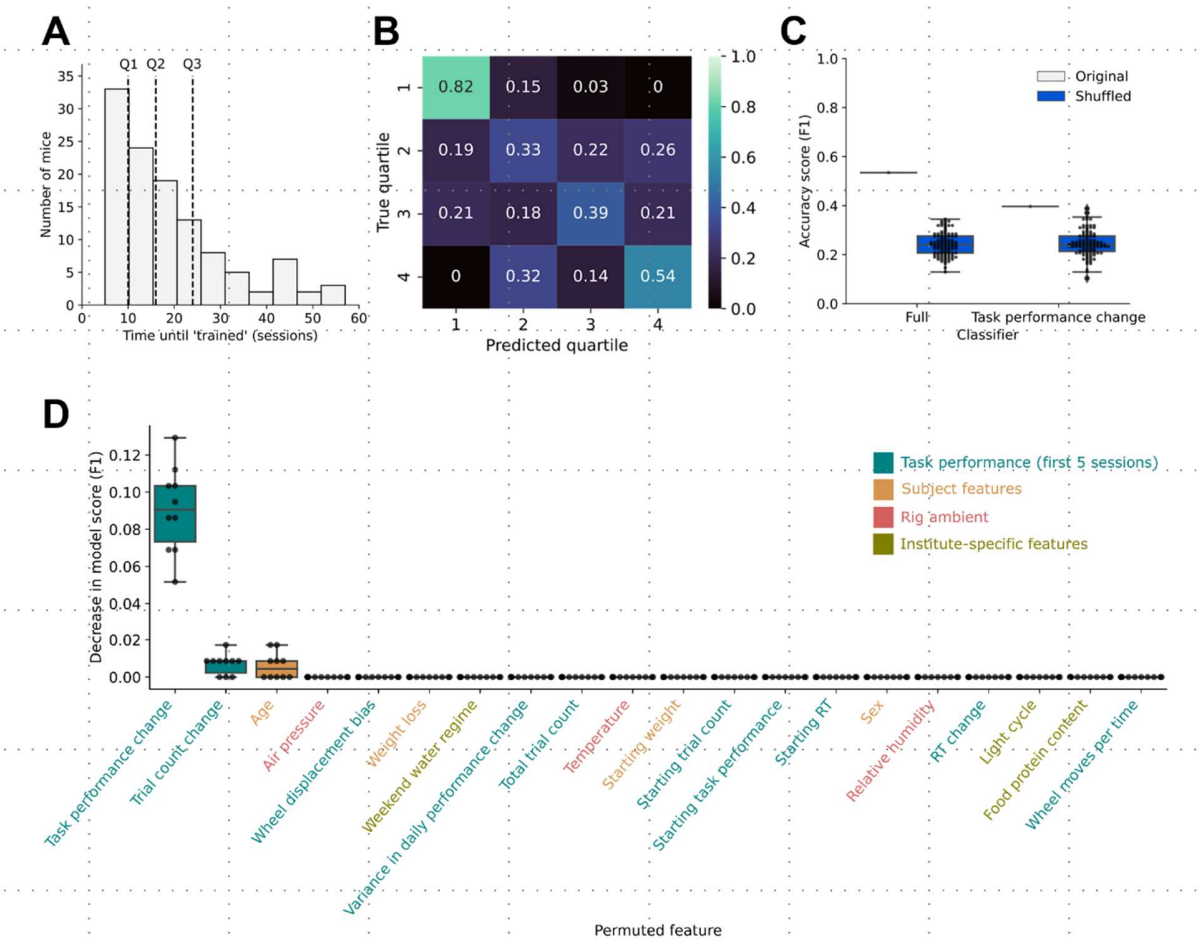

Figure 2 - *Predicting time taken to complete training from diverse data and metadata.* ***A.*** *Histogram of the number of training sessions taken to reach the IBL 'trained' criterion (N=116 mice). Vertical dashed lines represent the split of the data in quartiles.* ***B.*** *Cross-validated confusion matrix of a random forest classifier, trained to predict training time quantile from multiple behavioral features. Rows represent the true quartile and columns represent the predicted quartile; results were normalized over the number of mice of the corresponding true quartile (row).* ***C.*** *Prediction accuracy for a classifier that uses all features (Full classifier), and a classifier that uses only task performance change across the first 5 training sessions (Task performance change classifier). Horizontal lines show classifier performance; boxplots show distribution of performance scores over random shuffles of the training-time labels (N=100 shuffles).* ***D.*** *Importance of each feature in predicting training time. Box plots show the distribution of importance scores obtained across multiple permutations (N=10 permutations). In all boxplots, the box shows median and interquartile range, whiskers show range, points show individual observations.*

Supplementary Note 1: Lossless compression algorithm

We have developed a lossless compression algorithm for electrophysiology data, which achieves a 3-fold reduction in file size. The software is called Mtscomp and can be found at https://github.com/int-brain-lab/mtscomp. This algorithm provides lossless compression, and also random-access to quickly load small segments of the data without needing to decompress the entire files.

To do this, we took advantage of the temporal correlations in electrophysiological recordings, which show an approximate 1/f power spectrum. The input to the algorithm is represented as a flat binary multiplexed file of 2-byte integers, as typically produced by neurophysiology recording software. Data are compressed independently in consecutive chunks of one second, which allows random access to any part of the recording without decompressing the whole signal. To compress a chunk, we first compute discrete time differences independently for each channel, which approximately whitens the signal. We then compress the result using the zlib lossless compression algorithm. The initial values for each chunk and compressed difference signals are then appended to a compressed binary file on a chunk-by-chunk basis, and a companion JSON file stores the byte offset of every chunk. A decompression algorithm reads the JSON and binary files, allowing random “slices” of the data to be retrieved on the fly without decompressing the whole file. The compression code is unit-tested with 100% coverage. In our benchmarking we could achieve a ~3x compression ratio of our data. For our ~400 channel recordings, compression is ~4x faster and decompression is ~3x faster than real time on an Intel i9 10-core computer. This compression algorithm could be used in other applications that rely on multichannel time series of approximate 1/f spectrum, within and beyond neuroscience.

Supplementary Note 2: ONE dataset types and files

The Open Neurophysiology Environment (ONE) defines a set of conventions for naming and organizing data files, which allow data to be shared between labs in a standardized manner. These files are also sometimes called “ALF” files, for historical reasons.

The files for each experiment are stored in a single directory. Each file has a 3-part name of the form `object.attribute.extension`. Each file stores a single “dataset”. Datasets are usually numeric arrays, but can be arrays of any dimensionality, lists of strings, movies, or arrays of structures. The `object` and `attribute` together define what information is stored in the file (the “dataset type”), while the `extension` defines the file’s physical format. One can use any physical file format, provided that the extension alone makes it clear how to load the file. We recommend .npy files for numerical arrays (a binary format that includes array sizes) and .tsv (tab-separated text) files for text. However, flat binary files are not recommended as they do not encode array shape, and comma-separated .csv files are not recommended as they can become confused by strings containing commas. For movies we recommend .mj2 files, which allow random access loading of individual frames.

The `object` and `attribute` together describe the data contained in each dataset. All datasets pertaining to a specific object must have the same number of rows - i.e. the same size of their leading dimension. For example, the file `clusters.waveforms.npy` contains a 3d numerical array of size `[nClusters, nTimepoints, nChannels]`, storing the mean waveform of spikes in each electrophysiological cluster for each channel and time, while `clusters.brainLocationAcronyms_ccf_2017.tsv` contains a 1d string array of size `[nClusters]` containing the inferred brain location of these clusters according to the 2017 Allen common coordinate framework. The leading dimension of both arrays is the number of clusters recorded.

The naming of ONE files allows encoding of cross-references between datasets. If the attribute of one dataset matches the object of another, this represents a cross-reference. For example, `spikes.clusters.npy` contains an integer cluster assignment for each spike, which can be used to index all datasets in whose filename object is `clusters`. Thus, if the n^{th} row of `spikes.clusters` contains the integer m, then the n^{th} recorded spike was assigned to cluster m, and its waveform can therefore be looked up in the m^{th} row of `clusters.waveforms.npy`. These cross-references must use Python/C conventions, with the first row having index 0.

Several other conventions apply to ONE data. For example any dataset type whose attribute is of the form `times` or `*_times` represent the times of events, measured in seconds relative to experiment start. If data from multiple devices needs to be time-synchronized, this must be done before producing these `times` datasets, which are thus all on a comparable timescale. Datasets whose attribute is `intervals` or `*_intervals` are two-column arrays giving start and end times of particular events in the same timescale of seconds relative to experiment start. Datasets have specified measurement units. For example, locations in the brain are given in Allen CCF

coordinates[1], measured in mm. The units of measurement for all dataset types are specified in the documentation.

All files for an experiment are stored in the same directory, with directories organized by subject name,experiment date in ISO format and experiment number. For example, the files of the day's first experiment for the subject "Hercules", collected on 1 June, 2022 would be stored in the directory `/Hercules/2022-06-01/001`. Further subdirectories, known as "Collections", provide a way to encode multiple datasets of the same type. For example there may be multiple `spikes.times` datasets representing recordings from different probes. These are organized in subdirectories of the main experiment directory: e.g. `/Hercules/2022-06-01/001/probe00/spikes.times.npy` and `/Hercules/2022-06-01/001/probe01/spikes.times.npy`.

"Revisions" provide a way to store multiple versions of a dataset, for example following software updates. Revisions are also stored in subdirectories (including subdirectories of the collection if there is one), whose name starts and end with pound signs and is typically an ISO date, e.g. `#2021-07-13#`. For example, if spike sorting for probe 00 of the experiment conducted on June 1 in subject "Hercules" was revised on July 13, the results would be stored as `/Hercules/2022-06-01/001/probe00/#2022-07-13#/spikes.times.npy`.

ONE defines a list of standard dataset names. If data sharers use these names, it allows data users to understand the data without needing to read experiment-specific documentation. The names are listed in the table below in the form `object.attribute`. The extension, which specifies the physical format, is left to the data provider. Not all data can be standardized: many files will be specific to a particular experimental design. To enable both standardization across projects and extensibility, objects beginning with an underscore character are not expected to be standardized across projects, but objects not beginning with underscores are expected to be common across projects. The table below lists the dataset types currently used in IBL's implementation of the ONE standard. Datasets objects whose name does not begin with an underscore character contain data we believe can be standardized with many projects; these are largely adopted from the NWB data model. Those beginning with `_ibl_` contain data likely to be specific to our task or recording hardware. A live list of dataset types is linked at https://github.com/int-brain-lab/ONE/tree/main/docs.

ONE Dataset	Dimension	Description
spikes.times	[nspi]	Times of spikes (seconds, relative to experiment onset). Note this includes spikes from all probes, merged together.
spikes.clusters	[nspi]	Cluster assignments for each spike (integers counting from 0). Cluster assignment reflects the result of manual curation.

spikes.depths	[nspi]	Depth along probe of each spike (µm; computed from waveform center of mass). 0 means deepest site, positive means above this.
spikes.amps	[nspi]	Peak amplitude of each spike (µV).
spikes.templates	[nspi]	Template ID of each spike (i.e. output of automatic spike sorting prior to manual curation)
spikes.samples	[nspi]	Time of spikes, measured in units of samples in their own electrophysiology binary file.
template.amps	[ntemp]	Mean amplitude of each template (V)
templates.waveforms	[ntemp, nsw, nchSub]	Waveform of each template spike (stored as a sparse array, only for a subset of channels with large waveforms).
templates. waveformsChannels	[ntemp, nchSub]	Channels of the raw recording on which the template waveforms are defined.
clusters.uuids	[nc]	Unique identifier assigned to each cluster when ALF files created and during manual curation.
clusters.metrics	[nc, nmetrics]	Quality control metrics for each cluster.
clusters.mlapdv	[nc, 3]	Estimated coordinates of the cell relative to bregma (mm) sein Allen Common Coordinate Framework (CCF)[1].
clusters. brainLocationIds_ccf_2017	[nc, 1]	Brain location id of clusters following ephys alignment obtained from 25um resolution 2017 Allen Common Coordinate Framework
clusters. brainLocationAcronyms_ccf_2017	[nc, 1]	Brain location acronym of clusters following ephys alignment obtained from 25um resolution 2017 Allen Common Coordinate Framework
clusters.waveforms	[nc, nsw, nchSub]	Waveform from spike sorting templates (stored as a sparse array, only for a subset of channels closest to the peak channel)
clusters. waveformsChannels	[nc, nchSub]	Identities of the channels that are represented in clusters.waveforms for each cluster sorted by amplitude.
clusters.depths	[nc]	Depth of mean cluster waveform on probe (µm). 0 means deepest site, positive means above this.
clusters.peakToThrough	[nc]	Trough to peak time of mean cluster waveform (ms).
clusters.amps	[nc]	Mean amplitude of each cluster (V)
clusters.channels	[nc]	Channel which has the largest amplitude for this cluster.
clusters.probes	[nc, np]	Which probe this cluster came from (counting from zero).
probes.trajectory	[np, 7]	Trajectory coordinates of probe

probes.description	[np]	Text description of probe: label (folder name), Model (3A, 3B1, 3B2), Serial Number, Original file name.
channels.probes	[nch]	Probe assignments for each channel (integers counting from 0). Can be used as direct indexing for the probes.* attributes.
channels.rawInd	[nch]	Array of indices in the raw recording file (of its home probe) that each channel corresponds to (counting from zero).
channels.mlapdv	[nch, 3]	Channel location relative to bregma (mm) in Allen CCF.
channels.localCoordinates	[nch, 2]	Location of each channel relative to probe coordinate system (μm): x (first) dimension is on the width of the shank; (y) is the depth where 0 is the deepest site, and positive above this.
eye.timestamps	[nEyeSamples, 2]	Timestamps for pupil tracking timeseries: 2 column array giving sample number and time in seconds.
eye.raw	[nEyeSamples, nX, nY]	Raw movie data for pupil tracking.
eye.area	[nEyeSamples]	Area of pupil (pixels^2).
eye.xyPos	[nEyeSamples, 2]	Matrix with 2 columns giving x and y position of pupil (in pixels).
eye.blink	[nEyeSamples]	Boolean array saying whether eye was blinking in each frame.
licks.times	[nLicks]	Times of licks is seconds.
spontaneous.intervals	[nSpontInt, 2]	Times when no other protocol was going on for at least 30 seconds.
_ibl_wheel.position	[nWheelSamples]	Absolute rotation of wheel (radians) where positive = CCW
_ibl_wheel.timestamps	[nWheelSamples, 2]	Times of position in absolute seconds from session start, non-evenly spaced
_ibl_wheel.velocity	[nWheelSamples]	Tangential velocity of the wheel (rad/s) where positive = CCW
_ibl_wheelMoves. intervals	[nWheelMoves,2]	2 column array of onset and offset times of detected wheel movements in seconds relative to session start.
_ibl_wheelMoves.type	[nWheelMoves]	String array containing classified type of movement ('CW', 'CCW', 'Flinch', 'Other').
_ibl_trials. firstMovement_times	[nTrials]	1D array of first movement times in absolute seconds. The first movement is defined as the move onset time of the first movement that has an amplidue > 1/3 * target threshold. Movements considered for a trial must have

		feedback time > onset < goCue - 0.2.
_ibl_trials.intervals	[nTrials,2]	Start (i.e. beginning of quiescent period) and end (i.e. end of iti) times of each trial in seconds relative to session start.
_ibl_trials.included	[nTrials]	Boolean array of which trials to include in analysis, chosen at experimenter discretion, e.g. by excluding the block of incorrect trials at the end of the session when the mouse has stopped.
_ibl_trials.repNum	[nTrials]	The trial repetition number, i.e. how many trials have been repeated on this side (counting from 1).
_ibl_trials.goCue_times	[nTrials]	Time of go cues in choiceworld - in absolute seconds from session start, rather than relative to trial onset **NOTE:** this is the time the sound is actually played.
_ibl_trials.goCueTrigger_times	[nTrials]	Time of go cues in choiceworld - in absolute seconds from session start, rather than relative to trial onset **NOTE**: this is the time the trigger command is sent.
_ibl_trials.response_times	[nTrials]	Time in seconds relative to session start when a response was recorded (end of the closed loop state in bpod).
_ibl_trials.choice	[nTrials]	The response ID: -1 (turn CCW), +1 (turn CW), or 0 (nogo)
_ibl_trials.stimOn_times	[nTrials]	Times of visual stimulus onset in seconds relative to session start.
_ibl_trials. stimOnTrigger_times	[nTrials]	Times of visual stimulus onset trigger command in seconds relative to session start.
_ibl_trials.contrastLeft	[nTrials]	Contrast of left-side stimulus (0-1, nan if stimulus is on the other side).
_ibl_trials.contrastRight	[nTrials]	Contrast of right-side stimulus (0-1, nan if stimulus is on the other side).
_ibl_trials.feedback_times	[nTrials]	Time of feedback delivery (reward or noise) in seconds relative to session start.
_ibl_trials.feedbackType	[nTrials]	Whether feedback is positive or negative (-1 for negative feedback, 1 for positive feedback).
_ibl_trials.rewardVolume	[nTrials]	Volume of reward given each trial (µl).
_ibl_trials.itiDuration	[nTrials]	Intertrial interval duration for each trial, from response time to end of trial (end of trial is beginning of quiescence period) this includes the feedback delivery and the 1or 2 seconds delay and the 0.5 sec iti at the end of each trial.
_ibl_trials.probabilityLeft	[nTrials]	Probability that the stimulus will be on the left hand side for the current block. The probability of right is 1 minus this

_ibl_passivePeriods.intervalsTable	[2, 4]	Intervals: choiceword / spont activity / RF mapping / replay task stim [start, end] times columns = ['passiveProtocol', 'spontaneousActivity', 'RFM', 'taskReplay'] lines = ['start', 'stop']
_ibl_passiveRFM.times	[nFrames]	passive RFM frame times
_ibl_passiveGabor.table	[nRepeats, 5]	Gabor patch presentations table columns = [start, stop, position, contrast, phase] lines = nPresentations = 180
_ibl_passiveStims.table	[nRepeats, 6]	All other stimuli times columns = [valveOn, valveOff, toneOn, toneOff, noiseOn, noiseOff] lines = nStims = 40
_iblrig_RFMapStim.raw	[nFrames, nx, ny]	RAW matrix RF mapping matrix (nframe_times, nx, ny)
camera.dlc	[nframes, npoints x 3]	Coordinates of DeepLabCut (DLC) points (x position, y position, likelihood). Total points = 11 (fpaws-2, nose-1, spout-2, tongue-2, eye-4).
camera.times	nframes	Time of each frame acquisition (training rigs: leftCamera; ephys rigs: leftCamera, rightCamera, and bodyCamera).
_iblqc_ephysTimeRms.timestamps	[ntwin]	Time scale for the RMS amplitude as a function of time, relative to the raw binary ephys file (s)
_iblqc_ephysTimeRms.rms	[ntwin, nch]	RMS amplitude as a function of time (V)
_iblqc_ephysSpectralDensity.freqs	[nfreqs]	Frequency scale for the spectrogram (Hz)
_iblqc_ephysSpectralDensity.power	[nfreqs, nch]	Spectral Density for all channels (V**2/Hz)
histology_3dimage.volume	[20, nx, ny,nz]	Raw histology imaging volume. Tiff file approximately 20 x [nx, ny, nz]
histology_3dimage.metadata	[n/a]	Histology imaging volume resampled or filtered.
histology_transform.elastix	[n/a]	Text file used by elastix to perform transform from a volume to another

The datasets “channels.mlapdv” and “clusters.mlapdv” define brain coordinates in 3 dimensions (mediolateral, anteroposterior, dorsoventral), relative to bregma defined as Voxel ML-566, AP-540, DV-33 within the 10µm volume of the Allen CCF mouse Atlas[1]. Mediolateral coordinates are positive for the right hemisphere; anteroposterior coordinates are positive for anterior; dorsoventral are positive for dorsal.

Supplementary Note 3: Open Neurophysiology Environment API

The Open Neurophysiology Environment (ONE) user interface allows users to search for experiments of interest and load data from them, without worrying about the format or location of the underlying files. This interface allows multiple backend instantiations, so users can run the same exact code to process data from multiple local or remote sources. We have provided two such instantiations: an "Alyx implementation" for large projects such as IBL, which requires a backend Alyx database; and a "local implementation" that allows data access to files on the user's local file system without an Alyx database. This local implementation allows data producers to release ONE-standardized data as a single zip file containing files in a variety of standard formats (npy, tsv, json, mj2, etc.), organized with one directory per experiment containing appropriately named data files.

The ONE API is implemented in Python. Full documentation is at https://one.internationalbrainlab.org, Below, we provide a brief summary of how to use it.

Setting up ONE

1. Installation

ONE can be installed as a standalone package with python 3.8 or later by running,

```
pip install ONE-api
```

2. Setup

To start using ONE, we must first configure some settings that tell ONE whether it should connect to a database or use a local file system, and which database to access. By default ONE is configured to connect to the public IBL database (which until curation is complete contains only a small number of experiments). This can be setup by typing the following

```
from one.api import ONE
one = ONE(silent=True, password='international')
```

Experiment IDs

ONE stores a collection of datasets for each experiment. Each experiment is uniquely identified by a string termed the experiment ID (eID), which for the Alyx implementation are UUID strings. One can find the `eIDs` of experiments matching desired criteria using the `one.search` command. For example, to find the eIDs of all experiments on the database conducted in the year 2020, one would type

```
eids = one.search(date_range=['2020-01-01', '2021-01-01'])
```

Further information on searching for experiments can be found by typing `help(one.search)` or `one.search_terms()`.

Datasets

The data associated with each experiment is collected in datasets, with two-part names of the form `object.attribute`, following the conventions described in Supplementary Note 2. The `extension` in the ONE files is hidden from users of the API, which returns the data as arrays, saving the user from needing to worry about the physical format.

To view the data available with the first eID in the list returned by the one.search command above, one would type:

```
one.list_datasets(eIDs[0])
```

To load an individual dataset one uses the `one.load_dataset` command. For example, to load spike times, the user would type:

```
st = one.load_dataset(eIDs[0], 'spikes.times')
```

To load all datasets belonging to an object, the command

```
spikes = one.load_object(eID, 'spikes')
```

will return a dictionary with one entry for each dataset associated with that object (`times, clusters, depths, amps`, etc.). Data from specific collections or revisions can be requested with optional arguments to the `one.load_*` commands.

In the Alyx version, data are downloaded from a remote server to a cache directory, then returned to the user. This means that the data only need to be downloaded from the server once.

Running without a database, or connecting to an alternative database

ONE can be used independently of a database by running from files in a local directory. This can be setup in the following way

```
from one.api import ONE

one = ONE(cache_dir='/home/user/downloads/ONE/behavior_paper')
```

To connect to a specific database other than the default, a base-url argument must be given. For example, to connect to a server from the (fictional) “mybrainlab” project one would type

```
from one.api import ONE
one = ONE(base_url='https://alyx.mybrainlab.org')
```

Data access from servers may be password protected and/or restricted to a whitelist of specific URLs; this feature is currently used for non-curated IBL data.

Supplementary Note 4: Comparison of data access protocols

There exist several ways to distribute and access neurophysiology data. Each has its own strengths, which are discussed below.

ONE comprises a set of conventions for naming the datasets associated with an experiment, and a lightweight API allowing users to search and load required data from these experiments. The user thus need not worry about underlying file formats or network connections, and data are cached on their local machine to avoid repeated downloads. Although format independent, we have favored widely supported data formats such as .npy, .tsv, and Parquet, allowing users to easily load data in the language of their choosing. ONE datasets are simple and readable: the file name and folder organization are descriptive, meaning users can work with the data without needing special loaders or preprocessing functions. ONE has additional features that make it suitable for a growing dataset with frequent contributions and diverse access needs. For example, ONE allows users to search and load specific data items, without downloading all data for an experiment, enabling analyses such as a comparison of behavior performance from all experiments to be performed quickly, without having to download the bulky physiology and video data. The versioning feature allows individual data items to be updated in real time without perturbing others, and allows users to "freeze" their analysis to the items available on a given date, for example when revising a paper. ONE's standardized data access functions allow conversion to other formats such as NWB (converter available at https://github.com/catalystneuro/IBL-to-nwb), and will also allow flexibility as neurodata standards evolve in the future. Finally, the local implementation of ONE allows single labs and collaborations to release ONE-standardized datasets with minimal effort, by naming files appropriately and "uploading and forgetting" on a website.

Neurodata without borders (NWB) was created by neurophysiologists and software developers to be a unified data standard suitable for diverse neurophysiological and behavioral data. NWB can store multimodal experimental data in a single file and thus is well suited for long-term distribution and storage of finalized data. Depending on an individual lab's needs, storing data in NWB format internally may further streamline data sharing: the metadata are stored in the same file as the data, so the full dataset is guaranteed to remain intact.

The Allen SDK is a library for accessing neurophysiology data produced by the Allen Institute for Brain Sciences. The ONE library has several different design features to this SDK. First, in ONE, dataset types are passed as string arguments rather than encoded in method names. Thus rather than running the AllenSDK command `data_set.get_spike_times()`, an ONE user would run the command `one.load_object(eID, 'spikes')`; this feature allows data producers to add new dataset types without needing to rewrite the API code, and thus will allow the API to be used by labs or collaborations other than IBL, who require different dataset types. Second, the ONE API allows users to access specific datasets from an experiment without downloading all data from that experiment in an NWB file; this feature allows users to quickly integrate data from

many experiments, for example the behavioral analyses reported in Ref. [2] could be produced without downloading the bulky electrophysiology data for each experiment.

DataJoint can be used to share data, especially for larger organizations, and was used to publicly release IBL's behaviour dataset. Datajoint is a framework not only for data release, but also for pipelined computation. It thus provides an opportunity to allow users to work on data in the cloud without explicit downloads, creating new pipelines for their own exploratory analysis; while powerful, this way of working is currently unfamiliar to many neurophysiologists. Hosting a DataJoint database costs money, and given the flexibility in user queries and computational resource use, resources must be closely monitored if these facilities are made widely available.

Supplementary Note 5: Auxiliary software

In developing the IBL data architecture, we have written several auxiliary open-source libraries that can be used by individual neurophysiology labs, collaborations large or small, as well as in fields beyond neuroscience.

Mtscomp (https://github.com/int-brain-lab/mtscomp) is a library that performs 3x lossless compression of raw neurophysiology data (Supplementary Note 1). This algorithm makes use of a statistical regularity in neurophysiology signals - their 1/f power spectrum - and thus could be used to compress data with similar properties in any field. The library allows users to extract data from the middle of a long recording without uncompressing the whole file, and thus also allows streaming random-access data from a remote server. As raw electrophysiology comprises the bulk of data stored required by IBL (and other similar projects), this provides a threefold saving on storage costs for such projects.

Interactive inspection and analysis of the large volumes of data acquired in the collaboration require effective visualization solutions. We have developed a toolbox, called Datoviz, that provides a unified GPU-based visualization platform for desktop applications combining 2D graphics, 3D objects, and graphical user interfaces. Datoviz is available at https://github.com/datoviz/datoviz, and is based on the Vulkan API, a successor to OpenGL that allows much faster dynamic visualization of large scientific datasets, in neurophysiology and beyond.

Finally, the code used to perform general neurophysiological data analyses are available in a growing toolbox termed Brainbox, which is written in Python and can operate independently of DataJoint and ONE. These tools are publicly available at https://github.com/int-brain-lab/ibllib/tree/master/brainbox, and are intended to be independent of the rest of the data architecture.

Supplementary Table 1: Resource list

i. Resources contributed through this paper

Resource	Description	Code	Documentation
Alyx	A user-friendly database for neuroscience data and colony management	https://github.com/cortex-lab/alyx	Install instructions: https://github.com/cortex-lab/alyx/#installation Usage: https://docs.google.com/document/d/1cx3XLZiZRh3lUzhhR_p65BggEqTKpXHUDkUDagvf9Kc/edit
mtscomp	A lossless compression scheme for electrophysiological data	https://github.com/int-brain-lab/mtscomp	https://github.com/int-brain-lab/mtscomp#multichannel-time-series-lossless-compression-in-python
ONE	A scheme for searching and loading ALF datasets	github.com/int-brain-lab/ONE	one.internationalbrainlab.org
Pykilosort	A python port of the Kilosort GPU spike sorting software, with template matching, clustering and drift correction	https://github.com/int-brain-lab/pykilosort/	https://github.com/int-brain-lab/pykilosort
brainbox	A python library of independent analysis functions oriented towards behavior and neurophysiology	github.com/int-brain-lab/ibllib	docs.internationalbrainlab.org/_autosummary/brainbox.html
Datoviz	A generic interactive data visualization library leveraging the graphics processing unit for high rendering performance	https://github.com/datoviz/datoviz	https://datoviz.org/
iblenv	A unified environment and issue tracker for all IBL Github repositories.	github.com/int-brain-lab/iblenv	github.com/int-brain-lab/iblenv#iblenv-installation-guide

LabCI	A small continuous integration server for remotely triggering tests in MATLAB and Python	github.com/cortex-lab/LabCI	github.com/cortex-lab/LabCI/#labci
IBL-to-NWB	A library for converting ONE files to NWB format	https://github.com/catalystneuro/IBL-to-nwb	https://github.com/catalystneuro/IBL-to-nwb#IBL-to-nwb

ii. Existing resources deployed as part of this architecture

Resource	**Description**	**Code**	**Documentation**
Neurodata Without Borders (NWB)	A data standard for neurophysiology that can be used to share, archive, use, and build analysis tools for neurophysiology data	pynwb.readthedocs.io/en/latest/getting_started.html#installation	pynwb.readthedocs.io/en/latest/api_docs.html
Kilosort2	A MATLAB toolbox for spike sorting on GPUs with template matching, clustering and drift correction	github.com/MouseLand/Kilosort	github.com/MouseLand/Kilosort/wiki
DataJoint	A relational data processing pipeline for the science lab in MATLAB and Python	github.com/datajoint Table implementation within IBL: github.com/int-brain-lab/ibl-pipeline	docs.datajoint.io/
Globus	An API for securely and robustly transferring large datasets between computers	globus.org/	docs.globus.org/
NPY-MATLAB	A small set of functions to read and write Numpy files in MATLAB	github.com/kwikteam/npy-matlab	github.com/kwikteam/npy-matlab#npy-matlab

Supplementary Table 2: Terminology

	What is it?	What is an example?	What is it for?
Bulk data	Large-scale raw recordings or derived preprocessed results.	Raw electrophysiology signal, spike sorting results, behavioral outcomes	Bulk data are processed, analyzed, and visualized to draw scientific conclusions.
Metadata	Small data items that provide information on bulk data.	Mouse strain/sex/lineage, electrode location, experimenter ID, recording hardware configuration and lab	Metadata allows users to search for bulk datasets they want to analyze, and is essential to interpret results.
Relational Database	A system that stores complex information in easily searchable and interdependent tables.	PostgreSQL, MySQL	Users search the database to find data they need. Searches are flexible, and can include queries not originally conceived of by the designers, eg, “Find all data run on male mice on a Tuesday”.
Data Standard	A set of rules specifying how data will be represented on a computer system.	Open Neurophysiology Environment, Neurodata Without Borders	A data standard allows users to read data from multiple providers without having to learn a new convention each time.
Data Pipeline	A software tool that allows complex multi-step computations to be run automatically.	DataJoint, joblib, Dask	Allows a standard analysis to be run repeatedly on multiple standardized datasets, saving intermediate results and continuing after interruptions.