The Journal of Open Source Software

Deployment strategies for scaling AutoLFADS to model neural population dynamics

Aashish N. Patel[1,2*], **Andrew Sadler**[3,4*], **Jingya Huang**[1], **Chethan Pandarinath**[3,4,5**], **and Vikash Gilja**[1**¶]

1 Department of Electrical and Computer Engineering, University of California San Diego **2** Institute for Neural Computation, University of California San Diego **3** Department of Biomedical Engineering, Georgia Institute of Technology **4** Center for Machine Learning, Georgia Institute of Technology **5** Department of Neurosurgery, Emory University ¶ Corresponding author * These authors contributed equally.

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Open Journals

Reviewers:

- @openjournals

Submitted: 01 January 1970
Published: unpublished

Summary

Advances in neural interface technology are facilitating parallel, high-dimensional time series measurements of the brain in action. A powerful strategy for interpreting these measurements is to apply unsupervised learning techniques to uncover lower-dimensional latent states and descriptions of their dynamics that account for much of the variability present in the high-dimensional measurements (Cunningham & Yu, 2014; Golub et al., 2018; Vyas et al., 2020). AutoLFADS (Keshtkaran et al., 2021) provides a novel deep learning approach for extracting estimates of these latent dynamics from neural population data. It extends the previously developed LFADS (Pandarinath et al., 2018) algorithm by leveraging population based training (PBT) [jaderberg2017population] to more effectively tune hyperparameters for accurate inference of latent dynamics. As hyperparameter sweeps are a computationally demanding process, the Ray library (Moritz et al., 2018) was initially used to enable processing across multiple machines. Recognizing differences in tooling across teams, research supercomputing, and commercial cloud providers, we extend available workflow options to efficiently leverage AutoLFADS on new datasets in various compute environments: local development in a container-native approach, unmanaged cluster development leveraging Ray, and managed cluster development leveraging KubeFlow and Kubernetes orchestration.

As modeling strategies employed by the neurosciences increasingly employ deep learning based architectures which require optimization of large sets of hyperparameters (Keshtkaran & Pandarinath, 2019; Willett et al., 2021; Yu et al., 2021), standardization and dissemination of computational methods becomes increasingly challenging. Although this work specifically provides an implementation using AutoLFADS, the tooling provided demonstrates strategies for employing computation at scale while facilitating dissemination and reproducibility.

Statement of need

Novel machine learning algorithms enable neuroscience researchers to uncover new insights on how the brain gives rise to behavior. In many cases, algorithms are developed with the intent for use across both current and future studies. However when new analysis algorithms are developed, their hyperparameters are often tuned to the datasets against which they were developed. This creates a significant barrier for new researchers and practitioners to evaluate and adopt an algorithm as they require finding suitable hyperparameters for their application. With the popularization of "AutoML" hyperparameter exploration libraries (HyperOpt, SkOpt, Ray), it is now possible to more extensively and effectively search the parameter space. Reducing the burden further and enabling scale, solutions like KubeFlow provide near codeless workflows

The Journal of Open Source Software

for evaluating algorithms and tuning models end-to-end. To facilitate the effective utilization and adoption of AutoLFADS, we provide three pathways to tackle the various use-cases that will empower users with local compute, users with access to ad-hoc or unmanaged compute, and users with access to managed or cloud compute.

Solutions

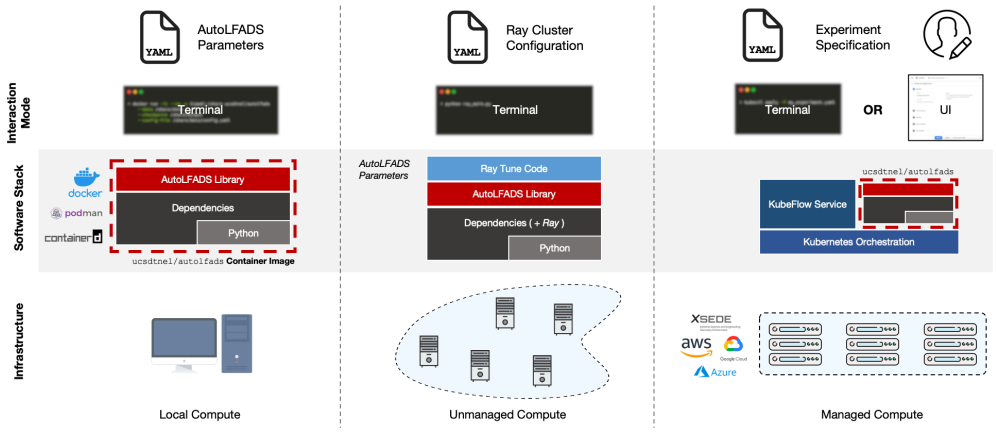

Figure 1: Solutions for running AutoLFADS on various compute setups. (Left) This column depicts a local workflow: users would leverage a container image that bundles all the AutoLFADS dependencies and provides an entrypoint directly to the LFADS package. Users can interact with this workflow by providing model configurations as a YAML configuration file and command line (CLI) arguments. (Middle) This column depicts a scalable solution using Ray: users would install AutoLFADS locally or in a virtual environment, provide model configurations and hyperparameter sweep specification directly in code, and then run a script providing the cluster configuration (network location and authentication) as a YAML file. (Right) This column depicts a scalable solution using KubeFlow: users provide an experiment specification that includes model configuration and hyperparameter sweep specifications either as a YAML file or using a code-less UI-based workflow. After experiment submission, the KubeFlow service spawns workers across the cluster that use the left column container images.

As users gain insight into novel datasets, it is often helpful to probe algorithmic parameters and investigate model performance locally. This can be accomplished by installing the LFADS package locally or in a virtual environment. Further isolating the workflow from local computational environments, we provide a pair of reference container images targeting CPU and GPU architectures where users can directly treat the image as an executable for which they simply need to provide the input neural data and desired LFADS model configuration. This approach has the benefit of eliminating the need for users to configure their environments with compatible interpreters and dependencies. Instead, the user installs a container runtime engine (e.g. Docker, Podman), which are generally well-supported cross-platform tools, to run the image based solution.

Scaling initial investigations may involve evaluating data on internal lab resources which may comprise a set of loosely connected compute devices. In such a heterogeneous environment, we leverage Ray to efficiently create processing jobs. In this approach Ray spawns a set of workers on the compute nodes that the primary spawner is then able to send jobs to. This approach requires users to provide a mapping of machine locations (e.g. IP, hostname) and

access credentials. While requiring users to manage the cluster lifecycle and artifacts produced, it provides useful flexibility beyond single node local compute.

To leverage large scale compute or managed infrastructure, we use KubeFlow which is an end-to-end machine learning solution that runs on top of Kubernetes based orchestration. In this arrangement, Kubernetes manages the underlying resource pool and is able to efficiently schedule compute jobs. Within KubeFlow, we leverage Katib (George et al., 2020) – KubeFlow's "AutoML" framework – to efficiently explore the hyperparameter space and specify individual trials. Furthermore, as KubeFlow is an industry-grade tool, many cloud providers offer KubeFlow as a service or provide supported pathways for deploying a KubeFlow cluster. Once deployed, KubeFlow enables configuration-based and code-less deployment of experiments.

Evaluation

One of the core innovations of AutoLFADS is its integration of PBT for hyperparameter exploration. Although less intensive than a full hyperparameter grid search, the depth of search is ultimately constrained by compute capacity. Thus, it is advantageous to run searches on remote compute solutions that facilitate rapid scaling. A comprehensive description of the AutoLFADs algorithm and results applying the algorithm to neural data using Ray can be found in Keshtkaran et al. (2021). To ensure that the KubeFlow based implementation can learn models of comparable quality, we trained a model using identical PBT hyperparameters, training data, and model configurations. We demonstrate comparable exploration of hyperparameters between Ray and KubeFlow in Figure 2, and similar converged model performances on metrics relevant to the quality of inferred latent dynamics in Figure 3 (Pei et al., 2021). This provides evidence that both implementations are converging to stable solutions given the stochastic behavior of PBT.

Table 1: AutoLFADS Performance. An evaluation of AutoLFADS performance on Ray and KubeFlow. Test trial performance comparison on four neurally relevant metrics for evaluating latent variable models: co-smoothing on held-out neurons (co-bps), hand trajectory decoding on held-out neurons (vel R2), match to peri-stimulus time histogram PSTH on held-out neurons (psth R2), forward prediction on held-in neurons (fp-bps). The trained models converge with less than 5% difference between the frameworks on the above metrics. The percent difference is calculated with respect to the Ray framework.

Framework	co-bps	vel R2	psth R2	fp-bps
Ray	0.3364	0.9097	0.6360	0.2349
KubeFlow	0.35103	0.9099	0.6339	0.2405
Percent difference	**+4.35**	**+0.03**	**-0.33**	**+2.38**

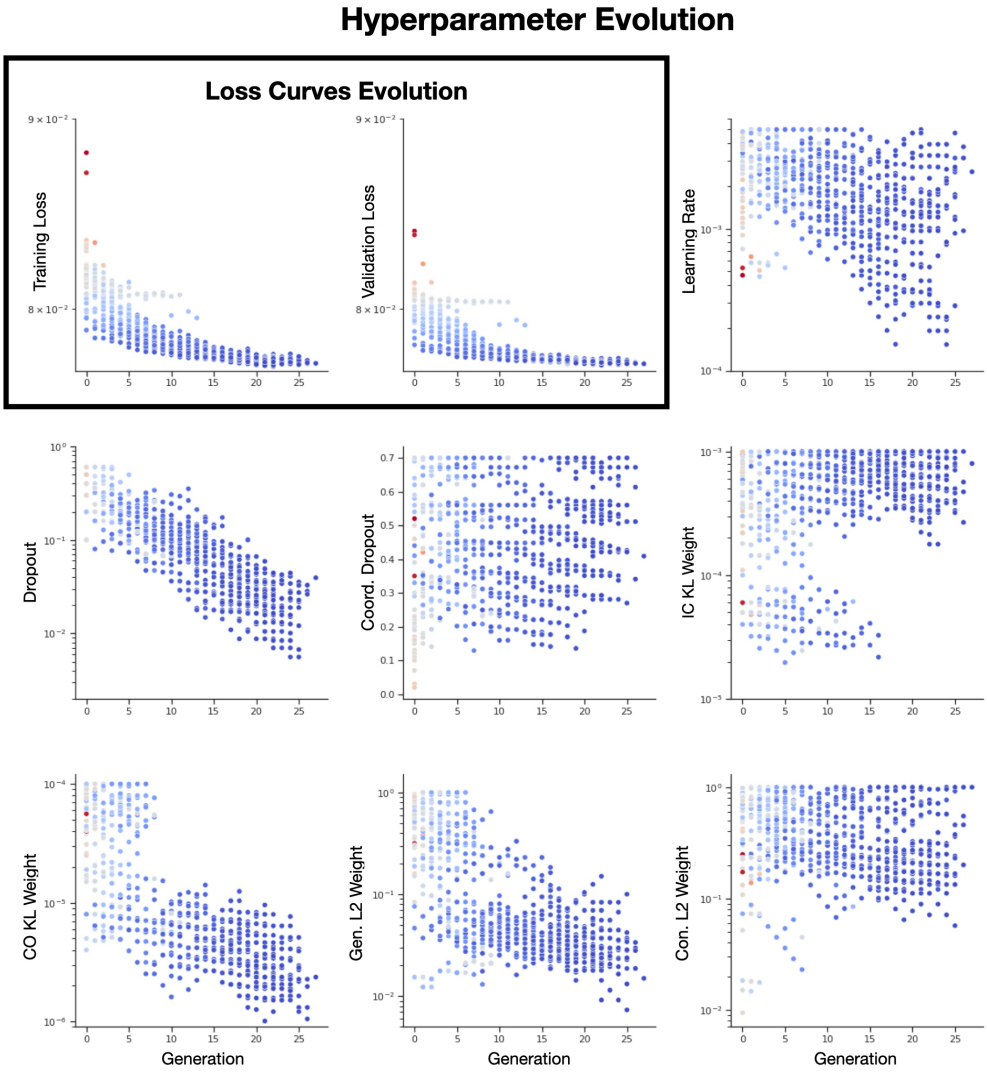

Figure 2: Evolution of loss curves and hyper-parameters for AutoLFADS using KubeFlow. Each point represents each individual training trial and darker blue colors represent models with better performance measured as lower validation loss.

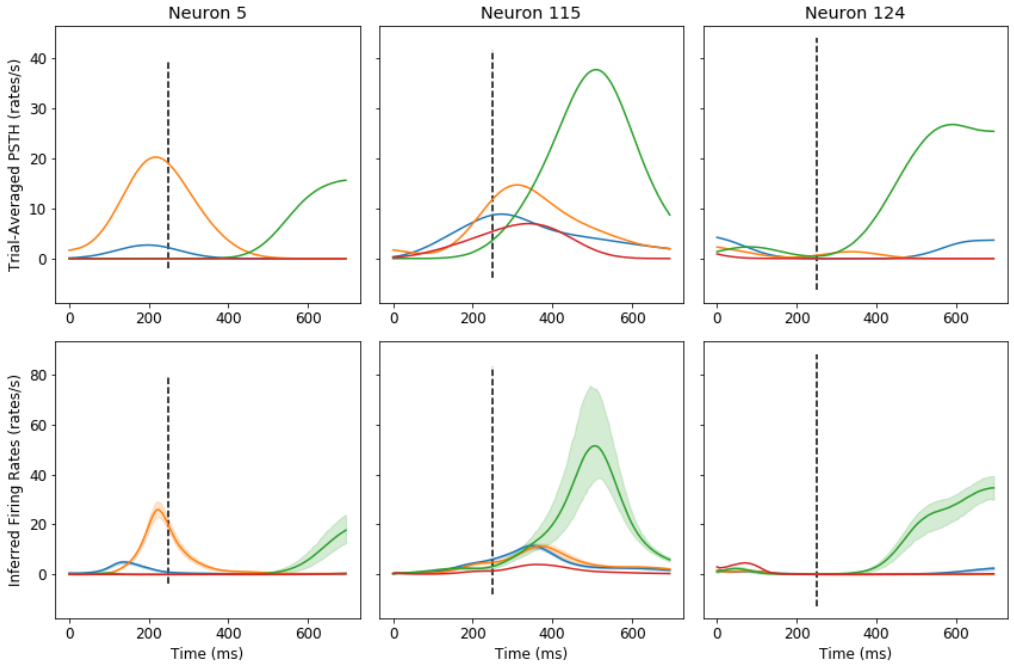

Figure 3: AutoLFADS inferred firing rates for three example neurons. Trial-averaged peristimulus time histogram (PSTHs) of recorded spikes (top) and inferred firing rates of AutoLFADS on KubeFlow (bottom) are time aligned to movement onset (dashed vertical line). Inferred firing firing rates (bottom row) depict averages across trials (dark line) and their standard error (thickness of translucent line).

References

Cunningham, J. P., & Yu, B. M. (2014). Dimensionality reduction for large-scale neural recordings. *Nature Neuroscience*, *17*(11), 1500–1509.

George, J., Gao, C., Liu, R., Liu, H. G., Tang, Y., Pydipaty, R., & Saha, A. K. (2020). *A scalable and cloud-native hyperparameter tuning system*. https://arxiv.org/abs/2006.02085

Golub, M. D., Sadtler, P. T., Oby, E. R., Quick, K. M., Ryu, S. I., Tyler-Kabara, E. C., Batista, A. P., Chase, S. M., & Yu, B. M. (2018). Learning by neural reassociation. *Nature Neuroscience*, *21*(4), 607–616.

Keshtkaran, M. R., & Pandarinath, C. (2019). Enabling hyperparameter optimization in sequential autoencoders for spiking neural data. *Advances in Neural Information Processing Systems*, *32*.

Keshtkaran, M. R., Sedler, A. R., Chowdhury, R. H., Tandon, R., Basrai, D., Nguyen, S. L., Sohn, H., Jazayeri, M., Miller, L. E., & Pandarinath, C. (2021). A large-scale neural network training framework for generalized estimation of single-trial population dynamics. *BioRxiv*.

Moritz, P., Nishihara, R., Wang, S., Tumanov, A., Liaw, R., Liang, E., Elibol, M., Yang, Z., Paul, W., Jordan, M. I., & others. (2018). Ray: A distributed framework for emerging {AI} applications. *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, 561–577.

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., & others. (2018). Inferring single-trial neural population dynamics using sequential auto-encoders. *Nature Methods*, *15*(10), 805–815.

Pei, F., Ye, J., Zoltowski, D., Zoltowski, D., Wu, A., Chowdhury, R., Sohn, H., ODoherty, J., Shenoy, K. V., Kaufman, M., Churchland, M., Jazayeri, M., Miller, L., Pillow, J., Park, I. M., Dyer, E., & Pandarinath, C. (2021). Neural latents benchmark '21: Evaluating latent variable models of neural population activity. In J. Vanschoren & S. Yeung (Eds.), *Proceedings of the neural information processing systems track on datasets and benchmarks* (Vol. 1). https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/file/979d472a84804b9f647bc185a877a8b5-Paper-round2.pdf

Vyas, S., Golub, M. D., Sussillo, D., & Shenoy, K. V. (2020). Computation through neural population dynamics. *Annual Review of Neuroscience*, *43*, 249.

Willett, F. R., Avansino, D. T., Hochberg, L. R., Henderson, J. M., & Shenoy, K. V. (2021). High-performance brain-to-text communication via handwriting. *Nature*, *593*(7858), 249–254.

Yu, X., Creamer, M. S., Randi, F., Sharma, A. K., Linderman, S. W., & Leifer, A. M. (2021). Fast deep neural correspondence for tracking and identifying neurons in c. Elegans using semi-synthetic training. *Elife*, *10*, e66410.