

## Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to **[Yes]**, **[No]**, or **[N/A]**. You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? **[Yes]** See Section ??.
- Did you include the license to the code and datasets? **[No]** The code and the data are proprietary.
- Did you include the license to the code and datasets? **[N/A]**

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
  - (a) Do the main claims made in the abstract and introduction accurately reflect the paper’s contributions and scope? **[Yes]**
  - (b) Did you describe the limitations of your work? **[Yes]** See Page 10
  - (c) Did you discuss any potential negative societal impacts of your work? **[Yes]**
  - (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? **[Yes]**
2. If you are including theoretical results...
  - (a) Did you state the full set of assumptions of all theoretical results? **[N/A]** The benchmark does not contain any new theoretical results.
  - (b) Did you include complete proofs of all theoretical results? **[N/A]**
3. If you ran experiments (e.g. for benchmarks)...
  - (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? **[Yes]** See Appendix A.3 and A.4
  - (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? **[Yes]** See Appendix A.4
  - (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? **[Yes]**
  - (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? **[Yes]** See Appendix A.4
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
  - (a) If your work uses existing assets, did you cite the creators? **[Yes]** See Appendix A.3
  - (b) Did you mention the license of the assets? **[Yes]** See Appendix A.3
  - (c) Did you include any new assets either in the supplemental material or as a URL? **[Yes]** See Appendix A.3
  - (d) Did you discuss whether and how consent was obtained from people whose data you’re using/curating? **[Yes]** See Appendix A.3
  - (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? **[Yes]** See Appendix A.3
5. If you used crowdsourcing or conducted research with human subjects...
  - (a) Did you include the full text of instructions given to participants and screenshots, if applicable? **[N/A]**
  - (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? **[N/A]**
  - (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? **[N/A]**

## A Appendix

### A.1 Additional Analyses

#### A.1.1 Maze co-smoothing and decoding

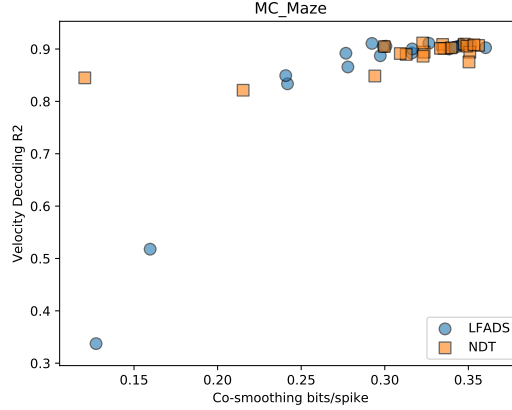
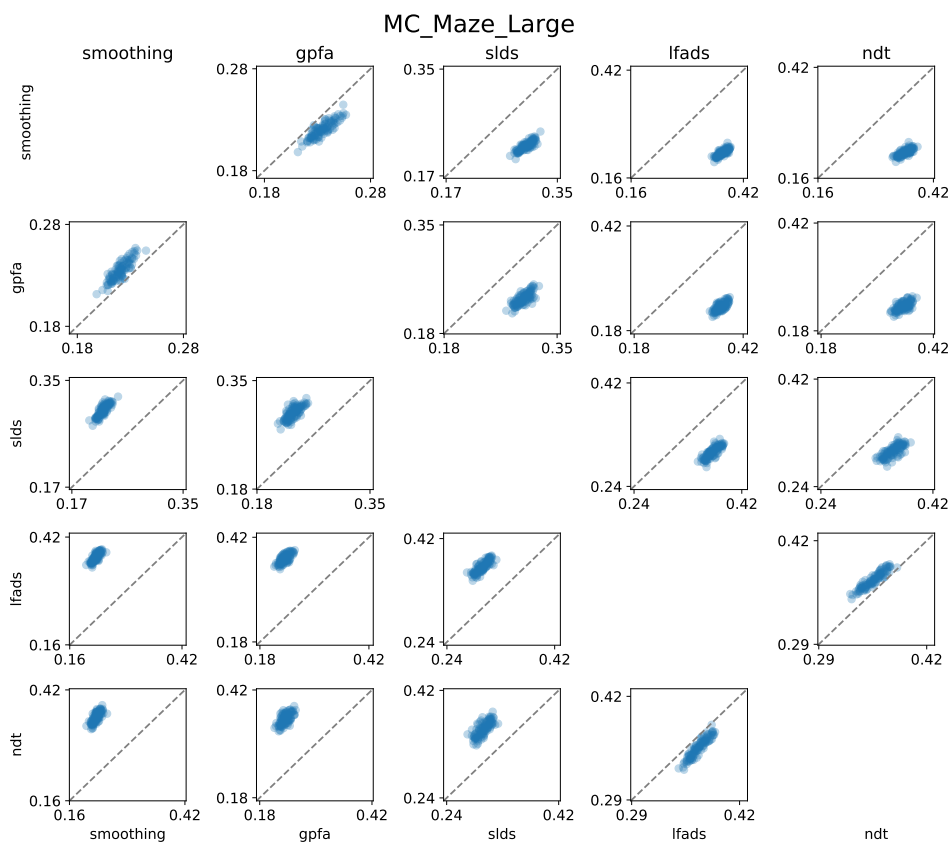
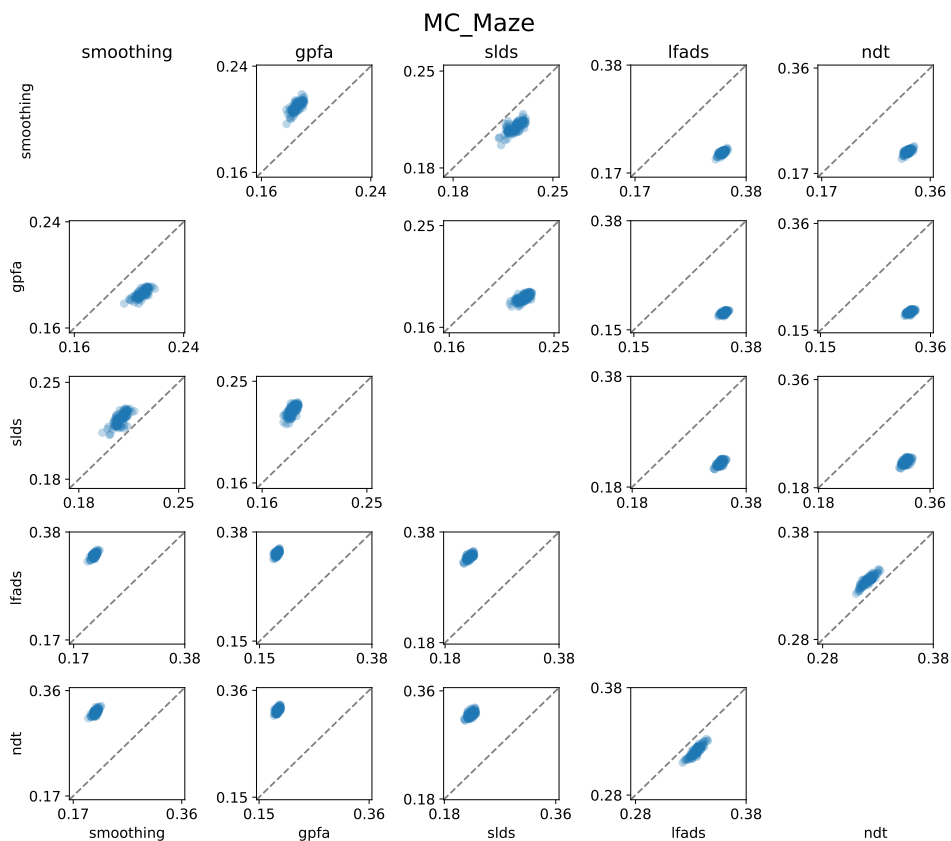


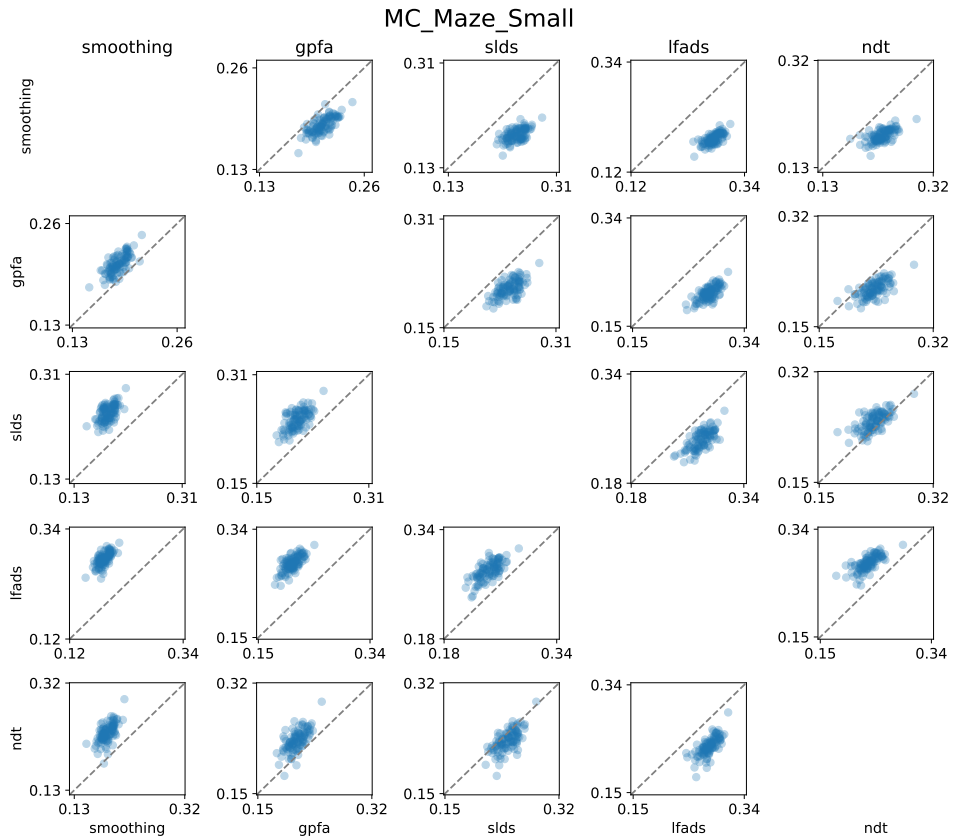
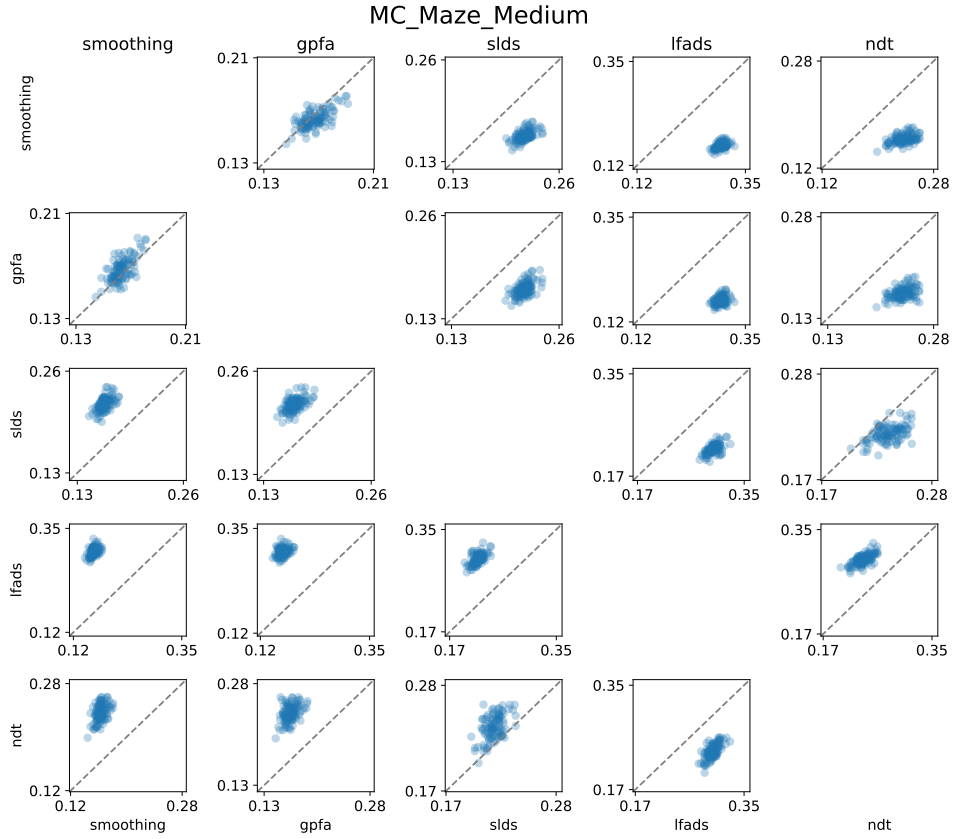
Figure 4: **Performance on co-smoothing and velocity decoding in MC\_Maze** . Random search LFADS and NDT models show good behavior decoding is possible under a wide range of fits to neural data, as quantified by co-smoothing performance.

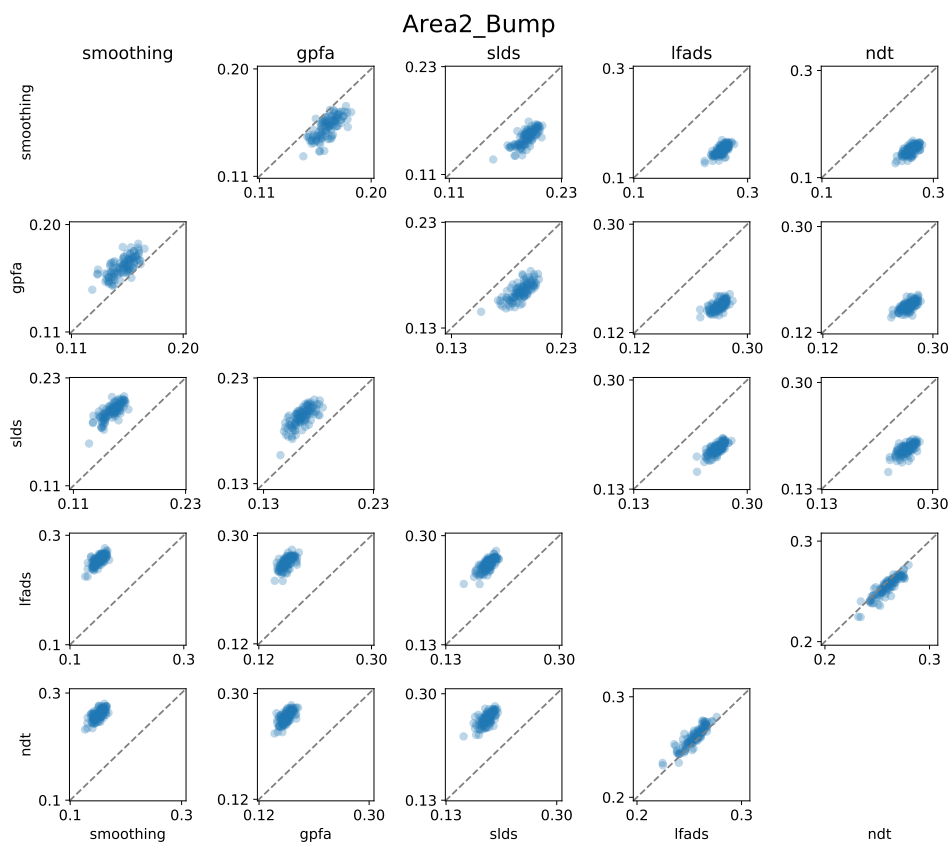
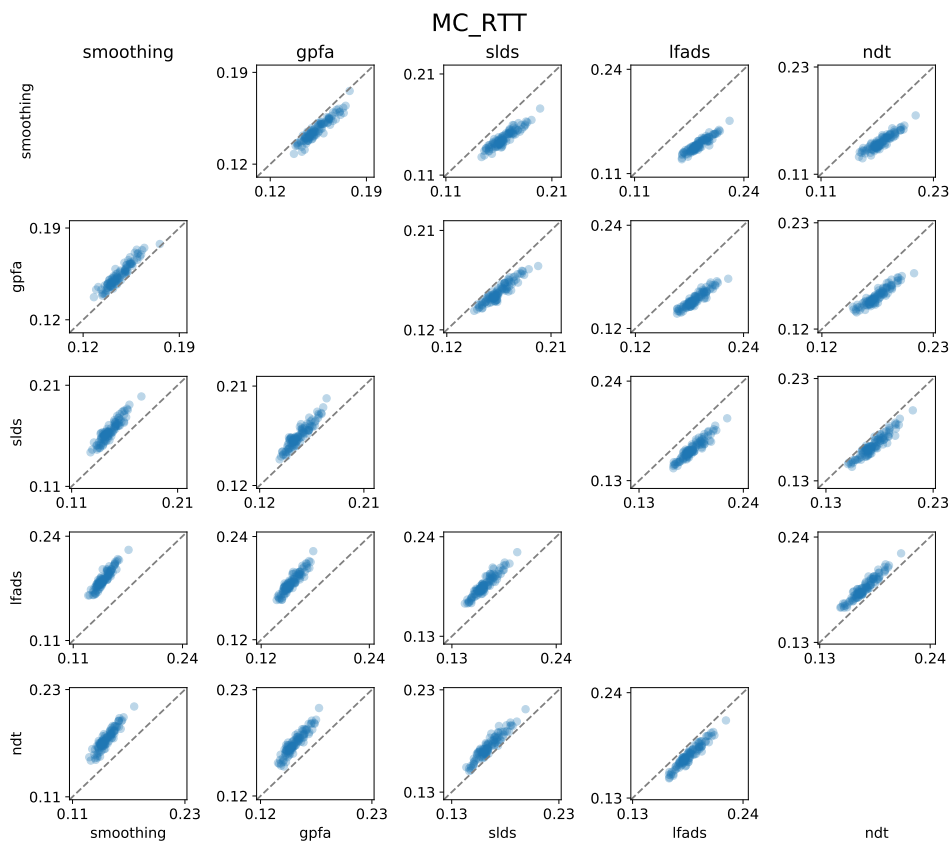
In order to examine the relationship between co-smoothing and behavioral decoding, we trained 20 LFADS and 20 NDT models with random parameters on the MC\_Maze dataset and evaluated their performance. As seen in Figure 4, both methods easily reach an apparent upper bound in decoding accuracy at around 0.9 but display a wide range of co-smoothing scores in that region. This demonstrates that there is an additional dimension to the models that is captured by co-smoothing but not by behavioral decoding.

#### A.1.2 Co-smoothing bootstrap analysis

To evaluate the reliability of the co-smoothing metric, we use a bootstrap analysis where we compare model performances pairwise on 100 bootstrap samples of the test set.







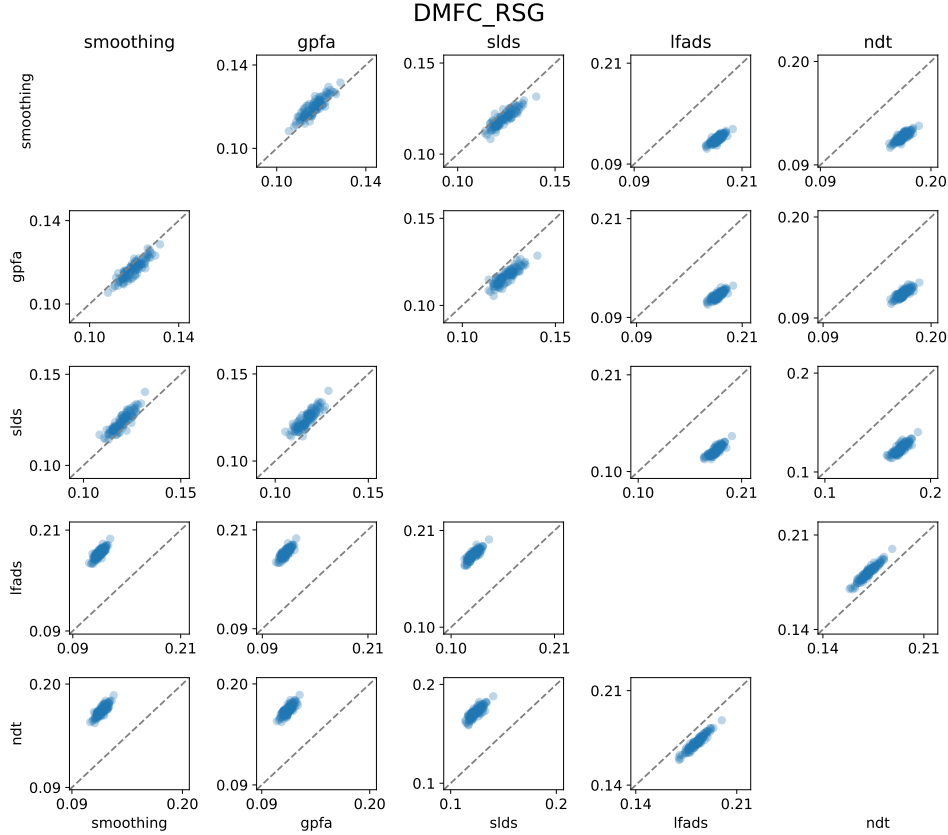


Figure 5: **Model co-smoothing performance compared across bootstrap samples.** Each panel compares the performance of two models, and within a panel, each point indicates the performance of the two models on the same bootstrap sample. The fact that most points lie on one half of the diagonal indicates that model rankings are robust to the particular evaluation dataset we used.

## A.2 The rapid growth of neural LVMs

As stated in our introduction, neural LVMs play an important role in analyzing the increasingly large neural datasets we can now record. We highlight in Table 3 related works that have been published in ML venues, to demonstrate the role the neuro-LVM sub-community plays within broader ML dialogue. Additionally, the table shows the wide variety of datasets and evaluation strategies used in these works. Note that even when the same dataset or metric is listed, very few implementations are consistently applied. For example, subsets of datasets are taken, metrics are normalized to different baselines, and model inputs and outputs vary. This provides a semi-quantitative account of the need for the NLB.

## A.3 Datasets

### A.3.1 Data format

The datasets have been converted to the Neurodata Without Borders (NWB) format [27], a standardized format for neurophysiological data. Neurodata Without Borders provides open-source Python and Matlab APIs for reading NWB datasets, which can be found from its GitHub organization at <https://github.com/NeurodataWithoutBorders>. The data format is based on HDF5, allowing any programming language with an HDF5 package to access the data as a typical HDF5 file. In addition, for our benchmark, we provide a code package that facilitates reading from our converted NWB files and extracting relevant data, available at [https://github.com/neurallatents/nlb\\_tools](https://github.com/neurallatents/nlb_tools).

Year	Venue	Work	Dataset	Held-out Channel	Behavior/ Stimulus	Metrics		
						PSTH	Held-out Time	Other
08	NeurIPS	Yu <i>et al.</i> [59]	[60]	✓				
11	NeurIPS	Macke <i>et al.</i> [9]	[60]	✓				
11	NeurIPS	Petreska <i>et al.</i> [61]	[60]		✓			
12	NeurIPS	Buesing <i>et al.</i> [62]	[60]	✓				
13	NeurIPS	Turaga <i>et al.</i> [63]	[63]				✓	✓
13	NeurIPS	Pfau <i>et al.</i> [64]	[65]					✓
14	NeurIPS	Semedo <i>et al.</i> [66]	[66]	✓				
15	NeurIPS	Gao <i>et al.</i> [67]	[60]	✓				
16	NeurIPS	Gao <i>et al.</i> [68]	[69, 60]			✓	✓	
17	NeurIPS	Nonnenmacher <i>et al.</i> [70]	[71]					✓
17	NeurIPS	Wu <i>et al.</i> [15]	[72]	✓	✓			
18	NeurIPS	Wu <i>et al.</i> [16]	[16]					✓
18	ICML	Duncker <i>et al.</i> [73]	[60]	✓	✓			
19	ICLR	Nassar <i>et al.</i> [54]	[69]		✓			
19	ICLR	Farschian <i>et al.</i> [74]	[74]		✓			
19	UAI	She, Wu [75]	[69]	✓				
19	NeurIPS	Lee <i>et al.</i> [76]	[77]		✓			
19	NeurIPS	Loaiza-Ganem <i>et al.</i> [78]	[60]	✓	✓	✓		
19	NeurIPS	Keshtkaran, Pandarinath [78]	[32]	✓	✓	✓		
19	NeurIPS	Schein <i>et al.</i> [79]	[80]					✓
20	ICML	Zoltowski <i>et al.</i> [55]	[81]	✓				
20	NeurIPS	Rutten <i>et al.</i> [82]	[31]	✓				
20	NeurIPS	Jensen <i>et al.</i> [83]	[84]	✓				
20	NeurIPS	Zhou, Wei [85]	[86]		✓	✓		
20	NeurIPS	Glaser <i>et al.</i> [87]	[86]				✓	
20	NeurIPS	Keeley <i>et al.</i> [88]	[69, 89]					✓
20	ICML	Keeley <i>et al.</i> [90]	[81, 89]					✓
21	ICML	Kim <i>et al.</i> [91]	[91]		✓			
21	NeurIPS	Liu <i>et al.</i> [92]	[77]		✓			✓
21	NeurIPS	Smith <i>et al.</i> [93]	[31]			✓		✓
21	NeurIPS	Zhu <i>et al.</i> [94]	[94, 28]	✓	✓			
21	NeurIPS	Jensen <i>et al.</i> [95]	[96]		✓			✓
21	NeurIPS	Hurwitz <i>et al.</i> [97]	[86]		✓			✓
21	NeurIPS	Bashiri <i>et al.</i> [98]	[98]	✓	✓			✓

Table 3: LVMs for neural data that have been published in ML venues in the last 15 years. Neural LVMs have become increasingly frequent but without any increased standardization. A check under a metric column indicates what data the model inference is evaluated against.

### A.3.2 Data hosting and licensing

The datasets are hosted on the platform DANDI (Distributed Archives for Neurophysiology Data Integration). DANDI is a platform specifically for publishing and sharing neurophysiology data. DANDI automatically generates structured metadata and persistent identifiers for the datasets uploaded to the site. The datasets are distributed under a Creative Commons Attribution 4.0 International license. The authors bear all responsibility in case of violation of rights. The datasets are available at the following links:

- MC\_Maze - <https://dandiarchive.org/dandiset/000128>
- MC\_RTT - <https://dandiarchive.org/dandiset/000129>
- Area2\_Bump - <https://dandiarchive.org/dandiset/000127/>
- DMFC\_RSG - <https://dandiarchive.org/dandiset/000130>
- MC\_Maze-L - <https://dandiarchive.org/dandiset/000138>
- MC\_Maze-M - <https://dandiarchive.org/dandiset/000139>
- MC\_Maze-S - <https://dandiarchive.org/dandiset/000140>

### A.3.3 Dataset documentation

All datasets curated for this benchmark have been featured in previous works. In the Source section for each dataset, we point to the original papers where one can find more detailed technical information on the datasets. Because each dataset features recordings from rhesus macaques, there is no personally identifying information or offensive content in the datasets.

Since all our datasets feature neural spiking activity from macaques, they share certain processing steps. In particular, all were spike sorted. Spike sorting is the process of identifying individual neurons, or units, from the voltage readings recorded by extracellular electrodes, which can pick up activity from multiple neurons. Spike sorting is an imperfect process, and multiple electrodes may pick up the same neural activity and sort it into separate units. This serves as the motivation for our removal of some sorted units based on cross-correlation.

Below, we document the source, our intended use, experimental design, data collection, and the additional processing we applied for each dataset.



## MC\_Maze

Name	Subject	Session date	Conditions	Training trials	Test trials	Held-in units	Held-out units
MC_Maze		2009-09-25	108	2295	574	137	45
MC_Maze-S	Name: Jenkins	2009-09-28	27	100	100	107	35
MC_Maze-M	Species: Macaca mulatta	2009-09-29	27	250	100	114	38
MC_Maze-L		2009-10-06	27	500	100	122	40

Table 4: **MC\_Maze summary.** Overview of assets included in MC\_Maze datasets.

**General description.** This dataset contains sorted unit spiking times and behavioral data from a macaque performing a delayed reaching task. The experimental task was a center-out reaching task with obstructing barriers forming a maze, resulting in a variety of straight and curved reaches. Neural activity was recorded from electrode arrays implanted in the motor cortex (M1) and dorsal premotor cortex (PMd). Cursor position, hand position, and eye position were also recorded during the experiment, and hand velocity was calculated offline from hand position.

**Source.** This dataset was collected by Matthew T. Kaufman and Mark M. Churchland from the Shenoy Lab at Stanford University. The dataset was created for the purpose of examining macaque neural activity during movement preparation and execution. The experiment and data collection is described in [28] and the dataset has been featured in a number of papers, including [41, 28, 31–40, 42, 8]. The dataset creators have granted permission to use and distribute the dataset sessions as part of the benchmark.

**Intended use.** This dataset has been curated for use in evaluating latent variable models of neural spiking activity as part of the Neural Latents Benchmark. The dynamics of the motor and premotor cortices during movement preparation and execution have been found to be well-modeled by autonomous dynamical systems. As such, the dataset tests how well models can infer such dynamics from noisy spiking activity. The inclusion of multiple sessions with varying trial counts also serves to evaluate how model performance scales with the amount of training data. The dataset is available on the platform DANDI to allow others to evaluate their methods on the data.

**Experiment design.** The maze task is a delayed-reach task. The task was performed with a cursor that was projected slightly above the monkey’s fingertips, which were tracked using a reflective bead. The task was performed in the plane of a vertical screen. The monkeys had to keep their hand close to the screen, but did not need to (and did not) slide along it.

In any particular trial, the monkey first fixated the eye and cursor inside a central fixation point. The monkey was then presented with a target and potentially barriers and distracters, which are unreachable targets that the monkey should ignore. The monkey maintained fixation until the go cue was signalled, indicated by the disappearance of the central fixation point and visual changes to the target. The monkey then made a rapid reach to the target and held at the target location for a short duration. If the cursor collided with a barrier before reaching the target, the display cleared and the trial ended.

The experimenters hand-designed 4 sets of 27 maze configurations. Each set of 27 maze configurations was grouped into 3 subsets of 3 mazes, each with 3 versions. The three versions of each maze have 1 target and no barriers, 1 target and barriers, or 1 target, 2 distracter targets, and barriers. Mazes within each subset were identical except for the placement of a few barriers directly obstructing the targets. In addition to hand-designed mazes, impossible mazes where no targets were initially reachable and randomly generated mazes were presented.

**Data collection methods.** Neural activity was recorded using two 96-electrode Utah arrays implanted in the primary motor (M1) and dorsal premotor (PMd) cortices. Recordings were sampled at 30 kHz and spike sorted offline. Sorted units were manually rated per trial by stability and cleanliness.

Target and go cue presentation time were recorded using a simultaneous flash of light out of the monkeys’ field of view that was detected using a “photo box”. Monkey movement onset time and reaction time were calculated offline using a robust algorithm.

Hand and eye tracking were used to record the monkey's movements and gaze. Cursor position was calculated real-time from hand position with various filtering and inertial prediction to make movement smoother. Values were recorded at 1 millisecond resolution.

**Processing.** For all sessions, trials with random or impossible mazes, unsuccessful reaches, or potential recording issues were discarded. Sorted units with low unit quality ratings or firing rates less than 0.1 Hz were removed. Cross-correlations for all pairs of sorted units were computed and neurons were removed until all cross-correlations were below a threshold. Thresholds were determined by plotting histograms of all cross-correlation values and identifying outliers. Neurons within each pair were removed based on which neuron was in more threshold-surpassing pairs, or, if that number was the same, by which had higher average cross-correlation with all other units. 25% of the remaining sorted units were randomly selected to be held-out in test data. A recording error in hand position was corrected by subtracting 8 mm from all  $y$  measurements. Hand velocity was estimated from hand position using second order accurate central differences. For MC\_Maze , 20% of the trials in each condition were randomly selected for the test set. For MC\_Maze-S , MC\_Maze-M , and MC\_Maze-L , trials were randomly selected from each condition for the train, val, and test sets. Ordering of trials in the train, val, and test sets was randomized for all sessions.

MC\_RTT

Name	Subject	Session date	Training trials	Test trials	Held-in units	Held-out units
MC_RTT	Name: Indy	2017-07-02	10.8 min	3.62 min	98	32
	Species: Macaca mulatta		1080 trials	271 trials		

Table 5: **MC\_RTT summary.** Overview of assets included in MC\_RTT dataset.

**General description.** This dataset contains sorted unit spiking times and behavioral data from a macaque performing a self-paced reaching task. In the experimental task, the subject reached between targets randomly selected from an 8x8 grid without gaps or pre-movement delay intervals. Neural activity was recorded from an electrode array implanted in the primary motor cortex. Finger position, cursor position, and target position were also recorded during the experiment.

**Source.** This dataset was collected by Joseph E. O’Doherty from the Sabes Lab at UCSF. The dataset was created for the purpose of examining neural activity during naturalistic self-paced motor behavior and evaluating the performance of behavioral decoding algorithms. Other sessions from this dataset and descriptions of the included data are available publicly on Zenodo [44]. The experimental procedure is described in detail in [99], from which the descriptions below are taken. The dataset creator has granted permission to use and distribute the dataset as part of the benchmark.

**Intended use.** This dataset has been prepared for use in evaluating latent variable models of neural spiking activity as part of the Neural Latents Benchmark. The unique task design results in the absence of pre-movement delay periods and repeated, highly constrained reach conditions typical of other reaching tasks. The ability to accurately model motor cortical activity during this more naturalistic behavior, and not only highly-stereotyped reaches with preparatory periods, is vital to developing effective BCI decoders and understanding the functioning of the motor cortex. The dataset is available on the platform DANDI to allow others to evaluate their methods on the data.

**Experiment design.** In the experiment, circular targets were presented in an 8-by-8 square grid. The monkey made movements in the horizontal plane and was shown the targets in a virtual reality environment using a mirror and projector. The monkey acquired targets by placing the fingertip of the working arm within a square acceptance zone centered on each target for 450ms. Targets were spaced such that acceptance zones were non-overlapping. After target acquisition, a new target was randomly drawn with replacement from the set of possible target locations and presented immediately. For a period of 200 ms after target acquisition, the next target was “locked-out” and could not be acquired.

**Data collection methods.** Neural activity was recorded using a single 96-electrode Utah array implanted in the primary motor cortex. Recordings were made using an RZ2 BioAmp Processor with PZ2 Preamplifier (TuckerDavis Technologies, Alachua, FL). The recordings were sampled at 24.4 kHz and filtered with a causal IIR bandpass filter (fourth-order Butterworth; Fpass = 500 Hz to 5000 Hz). Spikes were detected and sorted online using custom software.

Fingertip position was monitored with a six-axis electromagnetic position sensor (Polhemus Liberty, Colchester, VT) at 250 Hz. After acquisition, the position data were non-causally low-pass filtered to reject sensor noise (4th order Butterworth; Fcutoff = 10 Hz).

**Processing.** Unsorted “hash” units and sorted units with firing rates less than 0.1 Hz were removed. Cross-correlations for all pairs of sorted units were computed and neurons were removed until all cross-correlations were below a threshold. Thresholds were determined by plotting histograms of all cross-correlation values and identifying outliers. Neurons within each pair were removed based on which neuron was in more threshold-surpassing pairs, or, if that number was the same, by which had higher average cross-correlation with all other units. 25% of the remaining units were randomly selected to be held-out in test data. Finger velocity was estimated from finger position using second-order accurate central differences. The first 4.5 seconds and last 0.9 seconds of the continuous session were discarded as the subject was not actively performing reaches in those periods. The remaining continuous recording was divided into 9 segments alternating between test and train, resulting in 5 test segments and 4 train segments. The lengths of the test segments were randomly altered while

maintaining roughly equal train segments. The sizes of the test segments were limited to ensure they would collectively contain roughly 20% of the total “trials”, which were simply continuous 600 ms snippets of the recording. In the test set, an additional 200 ms of data was held-out per “trial” for forward prediction evaluation. Ordering of trials in the test set was randomized. The train segments were randomly divided into train and val splits.

## Area2\_Bump

Name	Subject	Session date	Conditions	Training trials	Test trials	Held-in units	Held-out units
	Name: Han						
Area2_Bump	Species: Macaca mulatta	2017-12-07	16	364	98	49	16

Table 6: **Area2\_Bump summary.** Overview of assets included in Area2\_Bump dataset.

**General description.** This dataset contains sorted unit spiking times and behavioral data from a macaque performing a reaching task with perturbations. In the experimental task, the subject performed delayed center-out reaches using a manipulandum to control a cursor. On a portion of the trials, the manipulandum applied a bump during the center hold prior to the reach. Neural activity was recorded from an electrode array implanted in the somatosensory area 2. Hand position, cursor position, force applied to the manipulandum, length and velocity of various arm muscles, and angle and velocity of various arm joints were all recorded during the experiment.

**Source.** This dataset was collected by Raeed Chowdhury from the Miller Lab at Northwestern University. The dataset was collected for the purpose of examining neural encoding of proprioceptive information. The experiment and data collection are described in [46], from which much of the information below was taken. The dataset creator has granted permission to use and distribute the dataset as part of the benchmark.

**Intended use.** This dataset has been prepared for use in evaluating latent variable models of neural spiking activity as part of the Neural Latents Benchmark. Area 2 is robustly driven by mechanical perturbations to the arm and has been shown to contain information about whole-arm kinematics. Thus, the experimental task results in both typical proprioceptive input and unexpected perturbations in the recorded brain area. As such, this dataset provides the challenge of modeling input-driven activity in response to both predictable and unpredictable sensory feedback. The dataset is available on the platform DANDI to allow others to evaluate their methods on the data.

**Experiment design.** In the experiment, the monkey performed a classic center-out reaching task. The monkey used a manipulandum to control a cursor to reach for targets presented on a screen in a 20 cm x 20 cm workspace. The experiment consisted of two types of trials: active and passive. In active trials, the monkey held the cursor in a target at the center of the workspace for a random amount of time, after which one of eight outer targets was presented. The monkey then reached toward the target, and the trial ended in success once the monkey reached the outer target. On passive trials, motors on the manipulandum delivered a 2 N perturbation to the monkey’s hand in one of the target directions during the center hold period. After the bump, the monkey returned to the center target, after which the trial proceeded like an active trial. Active and passive trials each made up 50% of the total trials.

**Data collection methods.** Neural activity was recorded from a 96-electrode Utah array implanted in the arm representation of area 2 of somatosensory cortex. The Cerebus recording system (Blackrock) was used to record neural data for the experiments. Signals were sampled at 30 kHz and threshold crossings were detected online. After data collection, the Plexon Offline Sorter was used to manually sort threshold crossings into putative single units.

The position of the handle was recorded using encoders on the manipulandum joints. The interaction forces between the monkey’s hand and the handle were recorded using a six-axis load cell mounted underneath the handle. Before each reaching experiment, markers of different colors were painted on the outside of the monkey’s arm. A custom motion tracking system built from a Microsoft Kinect was used to record the 3D locations of these markers with respect to the camera, synced in time to the other behavioral and neural data. Muscle length and velocity and joint angle and velocity were computed from the motion tracking data.

**Processing.** Sorted units with firing rates less than 0.1 Hz were removed. Cross-correlations for all pairs of sorted units were computed and neurons were removed until all cross-correlations were below a threshold. Thresholds were determined by plotting histograms of all cross-correlation values and identifying outliers. Neurons within each pair were removed based on which neuron was in

more threshold-surpassing pairs, or, if that number was the same, by which had higher average cross-correlation with all other units. The continuous recording was divided into 9 segments alternating between test and train, resulting in 5 test segments and 4 train segments. While the number of trials in train segments were kept roughly equal, the number of trials in test segments were randomly selected from the set of values that ensured there would be at least 4 trials of each condition in the test data. The total number of test trials was held to around 20% of the number of successful trials. Ordering of trials in the test set was randomized. The train segments were randomly divided into train and val splits.

Name	Subject	Session date	Conditions	Training trials	Test trials	Held-in units	Held-out units
Name: Haydn							
DMFC_RSG	Species: Macaca mulatta	2016-12-11	40	1006	283	40	14

Table 7: **DMFC\_RSG summary.** Overview of assets included in DMFC\_RSG dataset.

**General description.** This dataset contains sorted unit spiking times from a macaque performing a time-interval reproduction task. In the experimental task, the monkey was presented with two stimuli separated by a specific interval of time. The monkey then attempted to time their response such that the interval between the second stimulus and their response matched the interval separating the two stimuli. Neural activity was recorded from neural probes implanted in the dorsomedial frontal cortex.

**Source.** This dataset was collected by Hansem Sohn from the Jazayeri Lab at MIT. The dataset was collected for the purpose of examining the neural computations underlying Bayesian inference. The experiment and data collection are described in [49], from which much of the information below was taken. The dataset creator has granted permission to use and distribute the dataset session as part of the benchmark.

**Intended use.** This dataset has been prepared for use in evaluating latent variable models of neural spiking activity as part of the Neural Latents Benchmark. Recorded from a cognitive brain area, the dataset poses the challenge of modeling complex neural activity without a clear moment-by-moment behavioral correlate. The dataset is available on the platform DANDI to allow others to evaluate their methods on the data.

**Experiment design.** The Ready-Set-Go (RSG) task is a time-interval reproduction task. At the beginning of each trial, the monkey was presented with two fixation cues: a circle, indicating visual fixation on the display center, and a square, instructing the monkey to hold the joystick in its central position. While fixating, two visual flashes – Ready followed by Set – provided the first two beats of an isochronous rhythm. The monkey estimated the sample interval,  $t_s$ , between Ready and Set (i.e., estimation epoch), and used this information in the subsequent production epoch to generate the omitted third beat (Go) by initiating a response action. Monkeys received reward if the produced interval,  $t_p$ , between Set and Go was sufficiently close to  $t_s$ . The Go response action was an eye saccade or joystick movement to the right or left. The response modality was indicated by the color of the fixation cues. The response direction was indicated by the placement of the displayed target.

The sample interval  $t_s$  was sampled from one of two prior distributions, a ‘Short’ prior ranging between 480 and 800 ms, and a ‘Long’ prior ranging between 800 and 1200 ms. The full experiment consisted of 40 conditions: 5  $t_s$  values each for the two priors (‘Short’ and ‘Long’), two response modalities (eye saccade and joystick movement), and two target directions. Like the response modality, the prior condition was cued explicitly by the color of the fixation cues. Response modality and prior condition were varied in blocks of trials. The target direction was chosen randomly across trials.

**Data collection methods.** Neural activity was recorded with 3 Plexon probes implanted in the dorsomedial frontal cortex. Signals were amplified, bandpass filtered, sampled at 30 kHz, and saved using the CerePlex data acquisition system (Blackrock Microsystems, UT). Spikes from single-units and multi-units were sorted offline using Kilosort software suites.

**Processing.** Due to recording instabilities, many sorted units had substantial changes in firing rate through the session. Firing rates for each unit were plotted and visually inspected, and units that dropped out for portions of the session were removed. The beginning and end of the session were also removed, as an excessive number of units showed instabilities in those periods. Remaining sorted units with firing rates less than 0.1 Hz were removed. Cross-correlations for all pairs of remaining sorted units were computed and neurons were removed until all cross-correlations were below a threshold. Thresholds were determined by plotting histograms of all cross-correlation values and identifying outliers. Neurons within each pair were removed based on which neuron was in more

threshold-surpassing pairs, or, if that number was the same, by which had higher average cross-correlation with all other units. The continuous recording was divided into 13 segments alternating between test and train, resulting in 7 test segments and 6 train segments. While the number of trials in train segments were kept roughly equal, the number of trials in test segments were randomly selected from the set of values that ensured there would be at least 4 trials of each condition in the test data. The total number of test trials was held to around 20% of the number of successful trials. Ordering of trials in the test set was randomized. The train segments were randomly divided into train and val splits.



#### A.4 Baselines

To seed our benchmark, we applied 5 baseline methods of varying complexity to all 7 datasets. Baselines were run 3 times with different random seeds, and mean scores  $\pm$  standard error of the mean were reported. Spike smoothing, which simply fits a GLM from smoothed spikes to held-out spiking activity, consistently converges on the same solution regardless of random initialization and thus was not run multiple times. Code for our implementations of each method are available in our GitHub repo: [https://github.com/neurallatents/nlb\\_tools](https://github.com/neurallatents/nlb_tools). The scripts used for each baseline, their dependencies, and parameter search ranges can be found in the `examples/baselines/` directory. We summarize our approaches below. Note that reported values for computation resources used are only for one baseline run, not all three.

##### Spike smoothing

**Implementation.** Held-in spiking activity was convolved with a Gaussian kernel. A Poisson GLM was fit from the logarithm of the smoothed spikes (with a small offset to prevent taking the log of 0) to held-out spiking activity for co-smoothing rate predictions.

**Parameter optimization.** Gaussian kernel standard deviation and Poisson GLM regularization penalty were optimized with a grid search. Models were validated with 5-fold cross-validation within the training data. The best set of parameters was used to train a new model on the entire training set and make predictions on the test set.

Dataset	GPU count	GPU type	Runtime
MC_Maze	0	NA	3.2 hrs
MC_Maze-S	0	NA	0.19 hrs
MC_Maze-M	0	NA	0.42 hrs
MC_Maze-L	0	NA	0.88 hrs
MC_RTT	0	NA	1.67 hrs
Area2_Bump	0	NA	0.30 hrs
DMFC_RSG	0	NA	0.62 hrs

Table 8: **Spike smoothing computational resources.** Resources used for spike smoothing parameter optimization.

**Code availability.** Spike smoothing was implemented using the standard Python libraries numpy, scipy, and scikit-learn.

##### GPFA

**Implementation.** GPFA was run using the elephant Python package. The fit GPFA model was then used to transform test held-in spiking data to latent factors. Linear regression was fit from latent factors to held-in rates. Non-positive held-in rates were rectified to a small positive value. A Poisson GLM was fit from held-in rate predictions to held-out spikes to generate held-out rate predictions.

**Parameter optimization.** Latent dimensionality and regularization penalties for both the linear regression and Poisson GLM were optimized through grid search. Models were validated with 3-fold cross-validation within the training data. The best set of parameters was used to train a new model on the entire training set and make predictions on the test set.

**Code availability.** In addition to standard Python libraries, our GPFA implementation uses the public package elephant, available at <https://github.com/NeuralEnsemble/elephant>.

##### SLDS

**Implementation.** We used a modified version of the Linderman Lab’s ssm package for our implementation of SLDS. SLDS was trained on spiking activity from all neurons for both required and forward prediction time steps. To generate test predictions, we approximated the posterior on held-in

Dataset	GPU count	GPU type	Runtime
MC_Maze	0	NA	35.7 hrs
MC_Maze-S	0	NA	1.25 hrs
MC_Maze-M	0	NA	3.0 hrs
MC_Maze-L	0	NA	8.8 hrs
MC_RTT	0	NA	9.44 hrs
Area2_Bump	0	NA	3.3 hrs
DMFC_RSG	0	NA	19.2 hrs

Table 9: **GPFA computational resources.** Resources used for GPFA parameter optimization.

test spiking activity, using masks to indicate the missing held-out test data. Using the approximated posteriors, rate predictions were generated by smoothing observations for both the train and test data.

**Parameter optimization.** Latent dimensionality, number of discrete states, and the dynamics regularization penalty were optimized with a random search. Due to the long training times of SLDS, especially on datasets with large numbers of trials, parameter combinations were trained and evaluated on two random samples of 100 training and 100 validation trials (or 50 training and 50 validation for the MC\_Maze-S ). Random searches were performed with 20 parameter combinations. Models were trained for 50 iterations during the random search. After the random search, the three parameter combinations with the best co-smoothing scores were trained on the full training data for 50 and 100 iterations. The best results out of these 6 models were selected. If performance was unsatisfactory after the random search, parameters were further hand-tuned based on trends in random search results. Due to our time constraints, it is likely that SLDS performance can still be improved over our results.

Dataset	GPU count	GPU type	Runtime
MC_Maze	0	NA	73.2 hrs
MC_Maze-S	0	NA	8.2 hrs
MC_Maze-M	0	NA	11.8 hrs
MC_Maze-L	0	NA	17.2 hrs
MC_RTT	0	NA	19.3 hrs
Area2_Bump	0	NA	12.4 hrs
DMFC_RSG	0	NA	40.1 hrs

Table 10: **SLDS computational resources.** Resources used for SLDS parameter optimization.

**Code availability.** Our modified version of ssm is available on GitHub: <https://github.com/felixp8/ssm>. In the modified package, parts of ssm were re-implemented using PyTorch to reduce runtime. The original ssm package is available on GitHub as well: <https://github.com/lindermanlab/ssm>.

## AutoLFADS

**Implementation.** AutoLFADS was run using SNEl’s internal Python implementation. The model architecture was modified to only take held-in spiking activity as input to the encoder while outputting held-in and held-out rate predictions (for both held-out units and held-out timepoints).

**Parameter optimization.** Parameter optimization is done in AutoLFADS through Population Based Training.

**Code availability.** Because [8] is still under review, SNEl will not publicly release the code right now. The code will be made available when the paper is published. A separate implementation of AutoLFADS is available at <https://github.com/snel-repo/autolfads>. However, this implementation does not contain the modifications applied for co-smoothing.

Dataset	GPU count	GPU type	Runtime
MC_Maze	10	Nvidia GeForce RTX 2080	4.66 hrs
MC_Maze-S	10	Nvidia GeForce RTX 2080	0.58 hrs
MC_Maze-M	10	Nvidia GeForce RTX 2080	0.97 hrs
MC_Maze-L	10	Nvidia GeForce RTX 2080	1.94 hrs
MC_RTT	10	Nvidia GeForce RTX 2080	1.5 hrs
Area2_Bump	10	Nvidia GeForce RTX 2080	0.49 hrs
DMFC_RSG	10	Nvidia GeForce RTX 2080	1.89 hrs

Table 11: **AutoLFADS computational resources.** Resources used for AutoLFADS parameter optimization.

## NDT

**Implementation.** NDT was run with the public implementation linked below. The model architecture was modified to treat held-out neurons and timesteps as additional masked elements to predict in every sample.

**Parameter optimization.** Parameter optimization is through random search in a predefined grid (see public config files).

Dataset	GPU count	GPU type	Runtime
MC_Maze	10	Nvidia GeForce RTX 2080	4.5 hrs
MC_Maze-S	10	Nvidia GeForce RTX 2080	0.5 hrs
MC_Maze-M	10	Nvidia GeForce RTX 2080	1.0 hrs
MC_Maze-L	10	Nvidia GeForce RTX 2080	2.0 hrs
MC_RTT	10	Nvidia GeForce RTX 2080	1.5 hrs
Area2_Bump	10	Nvidia GeForce RTX 2080	0.5 hrs
DMFC_RSG	10	Nvidia GeForce RTX 2080	2.0 hrs

Table 12: **NDT computational resources.** Resources used for NDT parameter optimization.

**Code availability.** The NDT code is publicly available at <https://github.com/snel-repo/neural-data-transformers>.

## A.5 Evaluation Parameters

A number of parameters are used in evaluation, namely trial alignment ranges, decoding lags, and PSTH kernel standard deviations. Below, we describe our processes for choosing these values for each dataset.

### A.5.1 Trial alignment

For MC\_Maze , MC\_Maze-L , MC\_Maze-M , and MC\_Maze-S , trials are aligned from 250 ms before to 450 ms after movement onset. This alignment was chosen such that each trial contained a portion of both the pre-movement preparatory period and the actual reach.

For MC\_RTT , trials are created by splitting the continuous data into 600 ms segments. This value was simply chosen as a fairly typical trial length. The ability to infer latents from segments of comparable or shorter length is essential for real-time decoding of neural signals.

For Area2\_Bump , trials are aligned from 100 ms before to 500 ms after movement onset, either after target onset for active trials or in response to the bump for passive trials. This alignment was chosen so that each trial contained the majority of the corrective movement or reach. For passive trials, the alignment interval also guarantees the inclusion of the bump within the interval.

For DMFC\_RSG , the trials are aligned from 1500 ms before up to the go response, with up to 300 ms of jitter added to the alignment point. This alignment was chosen to include the entire production epoch (between Set and Go) for all successful trials in all conditions. While the estimation epoch

(between Ready and Set) is also of interest, the combined length of the estimation and production epochs on the longest trials exceeds the length of the shortest trials. Taking only the production epoch guarantees that no segments will contain data from more than one trial.

### A.5.2 Decoding lag

Because signals take time to travel between the brain and peripheral nerves, external behavior is expected to lag behind the brain activity which drives it. Thus, to find the optimal time difference between neural activity and behavior for decoding, we searched across lag amounts by evaluating decoding with 5-fold cross validation. We ran this evaluation for all 5 baseline methods, choosing the lag value giving the highest mean result across all methods. The resulting values are:

- MC\_Maze : 100 ms
- MC\_RTT : 140 ms
- Area2\_Bump : -20 ms
- MC\_Maze-L : 120 ms
- MC\_Maze-M : 120 ms
- MC\_Maze-S : 120 ms

Note that Area2\_Bump is recorded from a sensory area, so behavior is expected to precede the corresponding neural activity, which is confirmed by the negative optimal lag value.

### A.5.3 PSTH kernel width

PSTH calculation typically involves two steps: convolution with a Gaussian kernel and averaging across trials. In order to choose a kernel standard deviation, we evaluated various kernel standard deviations using leave-one-out cross validation: For each trial, we computed PSTHs from all other trials in the same condition as the given trial. We then calculated the Poisson likelihood given that trial's spiking activity, using the PSTHs as the firing rates. The kernel standard deviation giving the highest mean likelihood across all trials was chosen for the PSTH metric. For MC\_Maze-L, though the optimal value was 40 ms, we instead use 50 ms in order to facilitate comparison with MC\_Maze-M and MC\_Maze-S, both of which have optimal values of 50 ms. The resulting PSTH kernel standard deviations are:

- MC\_Maze : 70 ms
- Area2\_Bump : 40 ms
- DMFC\_RSG : 70 ms
- MC\_Maze-L : 50 ms
- MC\_Maze-M : 50 ms
- MC\_Maze-S : 50 ms

## A.6 DMFC\_RSG Changes After Initial Release

During the NeurIPS rebuttal period, LFADS random searches on DMFC\_RSG revealed models that achieved near-perfect correlation scores while performing extremely poorly on co-smoothing. These models also scored better on match to PSTH than models that performed very well on co-smoothing. After a series of new analyses, we determined the sources of the issues and subsequently made changes to our evaluation methods, data preparation, and dataset.

First, we concluded that the original PSTH metric implementation, which evaluated each individual trial's match to the empirical PSTHs, was poorly suited to datasets with high single-trial variance in neural responses. For such datasets, like DMFC\_RSG, excellent match to the neural data is detrimental to performance on the PSTH metric. Thus, we altered our PSTH metric to instead average model rate predictions within each condition and compare those to the empirical PSTHs. The original implementation penalizes accurate representation of single-trial variance that deviates from PSTHs, while the new implementation, by averaging across trials, mitigates this effect.

Second, we determined that our original trial windows aligned to the go response were problematic. Instead of using the entire trial window, average neural speed is calculated from only the set-go window, the length of which is  $t_p$ , one of the variables in our correlation calculation. Our original trial alignment guaranteed that the go response occurred precisely at end of the trial window, which we found allowed a trivial high correlation score. Specifically, a model can output identical rate inferences in every trial, that increase in instantaneous neural speeds towards the end of the trial. The problematic alignment then includes more low-speed timesteps for longer trials, resulting in a lower average neural speed and a trivial correlation between neural speed and  $t_p$ . To remedy this, we introduced jitter to the alignment windows so that the go response does not always land exactly at the end of the trial window.

Finally, we found that a particular neuron dominated neural speed estimates. This neuron alone was sufficient to achieve excellent correlation scores, while its removal resulted in extremely poor scores for all models. This imbalance led us to change the dataset entirely for one recorded from a different session.