BRAINWIDEBENCH: BENCHMARKING LARGE-SCALE PRETRAINING AND ACROSS-ANIMAL TRANSFER IN MULTI-REGION NEURAL RECORDINGS

Anonymous authors
Paper under double-blind review

ABSTRACT

Advances in large-scale neural recording have made it possible to collect data across many animals and distributed brain regions, raising the question of whether this scale can be exploited to learn general-purpose neural representations transferable across diverse downstream tasks. Yet, progress toward this goal has been limited by fragmented evaluation protocols and a narrow focus on individual task domains. Here, we present `BrainWideBench`, a benchmark for evaluating across-animal transfer on multi-region neural recordings, built on the International Brain Laboratory Brainwide Map dataset of neural and behavioral recordings spanning 276 brain regions from 139 mice performing a sensory-guided decision-making task. The benchmark is organized around three complementary task suites that evaluate whether learned representations support downstream decoding of behavior, can predict masked or future neural activity, and can recover biologically meaningful anatomical organization. With this benchmark, we systematically evaluate pretraining methods across transfer settings, including finetuning on downstream objectives and zero-shot generalization to unseen animals. Our results confirm pretraining improves performance over matched single-session baselines, but we show current methods exhibit heterogeneity in transfer capabilities: gains depend strongly on the alignment between pretraining objectives and downstream tasks. No single approach performs uniformly well across all three suites, and most methods are designed to only address a subset of them. Together, these findings suggest that learning representations that jointly generalize across behavior, dynamics, and anatomy remains an open challenge. By providing a unified and reproducible evaluation suite, `BrainWideBench` establishes a framework for measuring progress toward general-purpose models of the mouse brain. Code is available at this link.

1 INTRODUCTION

Over the past decade, advances in large-scale neural recording technologies have fundamentally reshaped systems neuroscience. High-density electrophysiology (Jun et al., 2017; Steinmetz et al., 2021) now enables simultaneous recording of hundreds to thousands of individual neurons across multiple brain regions, producing brain-wide datasets up to single-cell, single-spike resolution across many animals performing complex behaviors (Steinmetz et al., 2019; IBL et al., 2025b; Findling et al., 2025). For the first time, it is possible to observe distributed neural computation across the brain at scale. These datasets promise insight into how perception, decision-making, memory, and action emerge from coordinated population dynamics spanning cortex, thalamus, hippocampus, and beyond. Yet the field remains constrained by analysis approaches that operate at the level of individual sessions or isolated regions, limiting our ability to study shared principles from the growing body of multi-animal recordings (Stringer & Pachitariu, 2024).

Foundation models offer a fundamentally different way to approach these datasets (Dyer & Richards, 2025; Bommasani et al., 2021). Rather than training separate models for each animal, region, or task, foundation models seek to learn shared representations across heterogeneous recordings by extracting patterns that recur across individuals and brain areas (Ye et al., 2025; Azabou et al., 2025; Zhang et al., 2025; Wang et al., 2025). In this view, each recording offers a partial observation of a broader underlying computational system. Scaling across animals and anatomical coverage may allow models

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

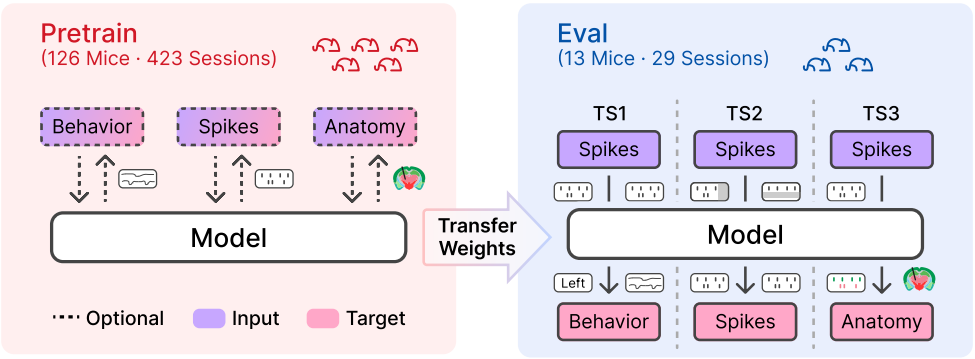


Figure 1: **Overview of the benchmark.** (Left) During pretraining, models have access to a large pretraining set with 126 mice that includes behavior, spike sorted neural recordings, and anatomical labels (e.g., brain region), to learn generalized representations. (Right) Pretrained models are evaluated on a held-out set of 13 animals across three task suites: TS1 (decoding), TS2 (neural activity prediction), and TS3 (brain region classification).

to distill invariant features of neural dynamics that are not visible within a single session or region. Similar scaling efforts in language and vision have revealed predictable trends in performance and emergent capabilities as data and model size increase (Radford et al., 2021; Hoffmann et al., 2022). Whether analogous principles hold for neural population activity, and whether large-scale training can uncover shared computational motifs across the brain, remains an open question.

Answering this question requires benchmarks that test whether learned structure extends to new neural populations (*across-animal transfer*) and whether representations remain useful beyond their training objectives (*cross-task generalization*). Existing benchmarks for spike-resolution recordings test at most one of these at a time. Some evaluate multiple objectives but hold out trials or stimulus conditions only within a session (Pei et al., 2021). Others test transfer across recordings but restrict evaluation to a single task family, typically behavioral decoding (Karpowicz et al., 2024; Willett et al., 2024) or stimulus encoding (Willeke et al., 2022). To our knowledge, no existing benchmark jointly tests both forms of generalization at the scale of multi-region recordings.

To address this gap, we introduce `BrainWideBench`, a benchmark for evaluating how large-scale pretrained neural models transfer across animals and downstream tasks. The benchmark comprises three task suites: behavior decoding, neural activity prediction, and brain region identification. `BrainWideBench` builds on the International Brain Laboratory Brainwide Map (IBL et al., 2025b), which spans over 600 hours of single-neuron recordings from 139 mice across 276 brain regions. To compare models consistently across heterogeneous recordings and prediction targets, we perform coordinated data curation. We develop a standardized quality control (QC) pipeline, adding new annotations for neural and behavioral signals, and establish shared protocols for pretraining and downstream evaluation. We pair a pretraining corpus of 126 mice with evaluation on 13 held-out mice, allowing us to assess how learned representations support the distinct tasks in unseen animals. We then use this framework to compare transformer- and state-space-based baselines, characterizing the strengths and limitations of current approaches to large-scale neural representation learning.

The main contributions of this work include:

- **A unified benchmark for transfer across animals and tasks**, organized around three complementary task suites: behavior and stimulus decoding, neural activity prediction, and anatomical identification. Shared evaluation protocols test whether pretrained representations capture structure that supports diverse downstream tasks in unseen animals.
- **A curated dataset derived from the IBL Brainwide Map**, with standardized pretraining and evaluation splits and newly annotated quality control (QC) metadata for neural and behavioral signals. These resources support reproducible comparisons and systematic analyses of how data scale, recording variability, and quality affect pretraining and transfer.
- **A rigorous empirical study of pretraining methods**, comparing supervised and self-supervised methods across architectures and training strategies, from single-session baselines to recent large-scale pretraining approaches. Evaluation across all three task suites reveals where pretraining improves transfer and where current methods remain limited.

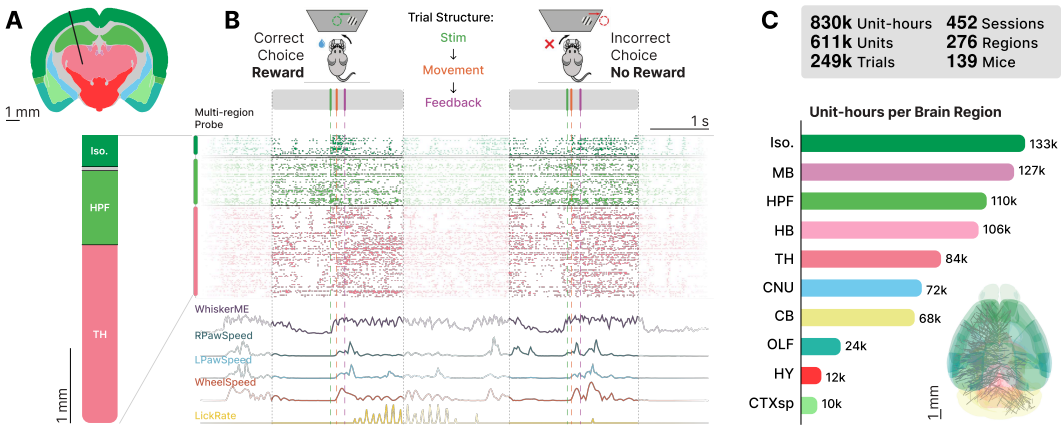


Figure 2: **Dataset overview.** (A) Multi-region probes are inserted into the brain and mapped onto a common coordinate framework (CCF) atlas to map individual neural spike sorted units to different anatomical regions. (B) An example of neural and behavioral data captured simultaneously while the mouse tries to control a wheel to move a visual stimulus from either the left or right side of the screen to the center. We capture paw and wheel speed, whisker motion, and licking from the water spout. After a correct trial, the mouse is rewarded with water. (C) Aggregated across probes and sessions, the dataset totals over 800,000 unit-hours.

2 Motivation

What are the goals of this benchmark? Large-scale modeling in neuroscience is rapidly expanding, with a growing number of works exploring pretraining and transfer for neural data (Azabou et al., 2024; Zhang et al., 2024; Ye et al., 2023; Willeke et al., 2022; Ryoo et al., 2025). These approaches have demonstrated promising gains across decoding, forecasting, and cross-session transfer, suggesting that neural activity contains substantial shared structure that can be leveraged through large-scale learning. However, evaluation remains fragmented: individual works often introduce new datasets, splits, modalities, or downstream tasks, making it difficult to determine whether improvements reflect genuinely more general representations or simply differences in evaluation protocols.

At the same time, an important open question remains: *what would a truly generalist neural foundation model look like?* Such a model should do more than solve a single decoding or forecasting problem. Instead, it should learn representations that support a diverse set of downstream objectives, including behavioral decoding, neural activity prediction, and anatomical identification, while generalizing across animals, recording sessions, and experimental conditions.

The goal of `BrainWideBench` is therefore two-fold: to provide a standardized and reproducible testbed for developing and comparing pretrained neural models, and to define a set of core downstream tasks that reflect fundamental properties of neural representations. To this end, `BrainWideBench` is organized into three complementary task suites. The first evaluates *behavioral decoding*, testing whether representations encode task-relevant variables that generalize across animals and sessions. The second measures *neural activity prediction*, assessing whether models capture structured spatiotemporal population dynamics. The third evaluates *neural identity prediction*, probing whether anatomical organization (brain region) emerges in pretrained representations and transfers without supervision. Together, these task suites test whether a single set of representations, pretrained across many mice and brain regions, transfers to held-out mice on multiple downstream tasks and under a standardized experimental setting.

Why this dataset? The International Brain Laboratory (IBL) Brainwide Map (IBL et al., 2025b) provides a rich multi-region dataset that is ideal for benchmarking pretraining strategies for multi-animal, large-scale neural recordings. It is a large-scale, multi-laboratory neurophysiology resource that captures brain-wide neural activity along with detailed behavioral measurements during a standardized decision-making task (Figure 2) (IBL et al., 2021). A key advantage of this dataset is its scale and consistency. Data were collected across a coordinated collaboration of laboratories using shared experimental protocols, hardware, and preprocessing pipelines, enabling reproducible comparisons across animals, sessions, and brain regions (Abbott et al., 2017; IBL et al., 2023; 2021;

2025a). This level of standardization is critical for evaluating generalization, as it reduces confounds introduced by dataset-specific variability.

The behavioral task involves visual decision-making, where mice respond to Gabor stimuli presented at varying contrasts and spatial locations. Structured manipulations of stimulus visibility and priors on spatial placement induce variability in perception, decision-making, and belief states, providing a rich substrate for probing behaviorally relevant representations. Neural activity is recorded using Neuropixels probes (Jun et al., 2017), yielding high-density spiking data from hundreds of neurons simultaneously. Across the dataset, recordings span 688 probe insertions in 139 mice, covering 276 anatomical regions and yielding over 600,000 units if no filters are applied (see Table A1 for unit counts under various filtering criteria). This brain-wide coverage enables evaluation of representations across diverse functional circuits, including sensory, motor, and cognitive systems.

In addition to neural recordings, the dataset includes synchronized behavioral signals such as trial structure, stimulus parameters, wheel movements, and video-based pose estimation (Biderman et al., 2024). Standardized preprocessing pipelines, including spike sorting (IBL et al., 2022b) and video analysis (IBL et al., 2022a), align these modalities into a unified format. The dataset is accessible through the Open Neurophysiology Environment (ONE) API (IBL et al., 2023), which provides a consistent interface for querying data across sessions, probes, and regions[1]. This enables scalable and reproducible benchmarking workflows. Together, the scale, standardization, multimodality, and brain-wide coverage of the IBL dataset make it particularly well suited for our benchmark.

3 BrainWideBench Task Suites

In the following sections, we detail the task suites that constitute the `BrainWideBench`, outlining motivations and specific design choices around them, as well as representative baselines. We characterize pretrained baselines in terms of their training objective: *task-suite-supervised* (`TSS`) models pretrain directly on the downstream tasks, and *task-suite-unsupervised* (`TSU`) models utilize a pretraining objective disjoint from the downstream tasks. To reduce complexity and standardize evaluation, models are tested on a single-session, single-task basis for each task suite. At the same time, users are free to explore different fine-tuning strategies, few-shot adaptation, or fully zero-shot transfer during evaluation. This design supports systematic study of scaling behavior, across-animal transfer, and representation learning in large-scale pretraining for neural data.

Across-animal transfer as a test of generalization. A central challenge in modeling neural data is learning representations that transfer across animals. Each recording provides only a partial and noisy view of a shared latent system underlying behavior and neural computation (Cunningham & Yu, 2014; Churchland et al., 2012). Given the broad anatomical coverage of the Brainwide Map (IBL et al., 2025b), successful transfer further requires models to capture neural structure and dynamics distributed across the brain. To evaluate this capability, `BrainWideBench` uses a strict animal-level split between pretraining and evaluation datasets (Appendix C), illustrated schematically in Figure 1.

This split structure enables evaluation of three complementary capabilities: (1) large-scale representation learning during pretraining, (2) adaptation to new animals with limited supervision, and (3) zero-shot generalization without task-specific fine-tuning. The pretraining corpus consists of recordings from 126 animals spanning 423 sessions and 274 brain regions, with access to all available supervision, including behavioral labels and anatomical annotations. Evaluation is performed on disjoint recordings from 13 held-out animals spanning 29 sessions and 124 regions, ensuring that models must generalize to unseen individuals rather than relying on subject-specific structure.

3.1 Task Suite 1: Behavior Prediction

TS1: Motivation. A central goal in systems neuroscience is to understand how neural activity gives rise to behavior (Glaser et al., 2020; Paninski et al., 2007). Behavior decoding provides a way to probe this relationship by evaluating how well models can extract behaviorally relevant information from neural population activity. From a neuroscience perspective, successful decoding enables the identification of behaviorally relevant signals distributed across populations, offering insights into

[1]Data is processed into the standardized benchmark format using `torch_brain` and released for download here. We also release the processing pipeline in our codebase for reproducibility.

how information is represented and transformed across circuits (IBL et al., 2021; 2025b). From an applications perspective, behavior decoding underpins brain-computer interfaces, neural prosthetics, and assistive technologies, where accurate inference of intent or state from neural activity is critical for real-time interaction (Pandarinath et al., 2018a; 2017; Karpowicz et al., 2024). In the context of large-scale pretraining, decoding tasks test whether pretrained representations generalize across animals, sessions, and tasks while preserving behaviorally meaningful structure (Azabou et al., 2024; Ye et al., 2025; Mao et al., 2026).

TS1: Tasks. This task suite includes eight behavioral decoding tasks derived from the Brainwide Map decision-making task (IBL et al., 2021). For each decoding task, the target is a 1 second window sampled around key trial events (i.e., stimulus onset, movement onset, feedback time) to ensure that there is both meaningful signal to decode the variable of interest and that we extract windows that are not confounded with other effects of movement (see Figure A9). In particular, we consider five *frame-level regression* tasks capturing continuous motor outputs: licking rate, whisker pad motion energy, wheel speed, left paw speed, and right paw speed (where wheel speed denotes the magnitude of angular velocity and paw speed the magnitude of 2D velocity); and three *sequence-level classification* tasks: stimulus contrast, choice, and reward. All of the continuous target variables are resampled at 50 Hz (details in Appendix B) and defined at each time point within the target window, producing a sequence-to-sequence prediction problem. See more details in Appendix D.1.

TS1: Evaluation Setup. At evaluation, each session (from an animal unseen during training) is split across time such that the first 40% of trials fall in the train segment, the next 20% are in validation, and the remaining 40% are in test. This temporal split serves two purposes: it eliminates autocorrelation leakage between segments, and it reflects a realistic deployment scenario in which a small portion of data is used to calibrate the model before generating predictions on the remainder. This setting is particularly challenging due to the long duration of recording sessions ($\sim$1.4h average over evaluation sessions), hence models must learn to deal with nonstationarities and shifting probes which can affect the quality of spike sorting. Final metrics are computed on the standardized test split. The train and validation splits are provided as a reference protocol, and we adopt supervised finetuning on all available data for our baseline models (see Appendix G for finetuning details of individual models). However, the benchmark is designed to support alternative evaluation regimes, including few-shot and zero-shot transfer, enabling future work to explore different adaptation strategies. We measure performance using R^2 for regression targets, balanced accuracy for classification targets, and D^2 under a Poisson model for Licking Rate (rate variable).

TS1: Baselines. We evaluate several pretrained models on TS1. We consider models to fall under the `TSS` setting if they pretrain on any of the decoding tasks, whereas `TSU` models utilize other forms of supervision or self-supervision during pretraining. In our framework, pretraining methods should produce a *single model* that can be applied downstream on all tasks in the suite. Hence, `TSS` models are naturally multi-task or otherwise assume cross-task transfer is achievable within the decoding targets; for this setting, we use POYO+ (Azabou et al., 2025) and the multi-task version of POSSM (Ryoo et al., 2025). For `TSU` models, we use NDT-Stitch (Ye et al., 2023; Zhang et al., 2024) and MtM (Zhang et al., 2024), which both perform masked neural prediction. Note that MtM also uses anatomical information in its masking scheme. In addition, we benchmark several representative architectures on a single-session, single-task basis with extensive hyperparameter tuning. These single-session baselines establish a strong reference point for the pretrained models on individual decoding tasks, and provide a basis for comparing architectures in a controlled setting. For these baselines, we use Linear, MLP, CNN (Lea et al., 2016), GRU (Cho et al., 2014), CEBRA (Schneider et al., 2023b), POYO (Azabou et al., 2024), and a supervised version of NDT (Ye & Pandarinath, 2021). See more details in Appendix F.

3.2 Task Suite 2: Neural Activity Prediction

TS2: Motivation. Neural systems operate through dynamic interactions across neurons, where patterns of activity propagate, transform, and ultimately give rise to behavior. Modeling these dynamics requires predicting how different neurons or brain regions shape each other's responses, making neural activity prediction a fundamental tool for capturing the temporal structure of population-level computation (Pandarinath et al., 2018a; Pei et al., 2021). By learning to predict neural activity, models can reveal the dependencies and interactions that govern circuit function, providing a direct lens on how information flows through the brain (Gokcen et al., 2022; Xia et al., 2026). Neural activity prediction thus serves as a testbed for *in silico* investigation of circuit mechanisms, enabling

the construction of surrogate dynamical systems and brain-scale simulators (Wang et al., 2025; Haspel et al., 2023) to probe hypotheses about circuit function. At the same time, generative modeling capabilities afforded by such prediction tasks support practical advances, including data augmentation to improve downstream analyses (Kapoor et al., 2024; McCart et al., 2024), enhanced interpretability through the discovery of latent structure in neural dynamics (e.g., Latent Variable Modeling (Pei et al., 2021; Hurwitz et al., 2021)), and real-time, closed-loop interfaces that adapt to evolving neural states (Minai et al., 2026).

TS2: Tasks. This task suite includes two neural activity prediction tasks designed to probe complementary aspects of neural representation along both spatial and temporal dimensions. First, to assess a model's ability to extract features of population-level structure and inter-neuron relationships, we design a *co-smoothing* task (Pei et al., 2021) where a random subset of neurons is masked within the context window, and the model must reconstruct their activity from the remaining observed neurons (Figure 1). At test time, the masked neurons form a fixed held-out set, ensuring consistent evaluation across samples. Next, to probe how well a model can extract fine-grained temporal structure from the population, we design a *forecasting* task with the goal of predicting future neural activity from past observations. This task assesses whether learned representations capture temporal dynamics and support extrapolation beyond the observed context. At test time, the last 100ms of each sample window are masked and set as the target for models to reconstruct. See more details in Appendix D.2.

TS2: Evaluation Setup. We find the causal train/validation/test splits of TS1 to be especially difficult for neural activity prediction in TS2, limiting our ability to differentiate models. A possible explanation is that non-stationarity in neural recordings makes neuron identity unreliable across long timescales, so predicting the activity of a fixed, indexed set of neurons becomes ill-defined (see Appendix I.2). Hence, in TS2, we adopt a split structure consisting of interleaved 5-minute temporal blocks, with 40% of the session in train, 20% in validation, and 40% in test. This block structure reduces autocorrelation leakage that would arise from fully randomized splits while ensuring that each subset is large enough to contain representative samples of neural dynamics (Appendix D.2). In both tasks, we evaluate performance using the fraction of deviance explained (D^2) under a Poisson observation model and bits-per-spike (bps) (Pei et al., 2021). D^2 can also be termed Deviance Fraction Explained (DFE) as in Xia et al. (2026). See more details on these metrics in Appendix E.

TS2: Baselines. In TS2, MtM (Zhang et al., 2024) serves as a representative `TSS` model since it explicitly trains on masked neural prediction objectives related to co-smoothing and forecasting. We also evaluate NDT-Stitch (Ye et al., 2023; Zhang et al., 2024), which pretrains using temporal masked prediction but in a non-causal setting, making it fall under `TSU` even with respect to forecasting. In addition, we include single-session baselines using a standard autoencoder (AE) (Hinton & Salakhutdinov, 2006), LFADS (Pandarinath et al., 2018a), and NDT (Ye & Pandarinath, 2021). Following the Neural Latents Benchmark (Pei et al., 2021), these models are treated as latent variable models (Hurwitz et al., 2021): models are fit to training data to learn population representations, which are then evaluated on co-smoothing and forecasting tasks. While TS2 also supports direct optimization on the downstream tasks themselves, we focus on representation quality as the primary evaluation objective. See Appendix F for additional details. We also test simple statistical baselines that use training-set neural activity statistics, providing reference points for how much performance can be explained without learning a representation (see Appendix I.2.1).

3.3 Task Suite 3: Neuron Identity Prediction

TS3: Motivation. A neuron's identity, including its anatomical location and cell type, strongly shapes its activity patterns across behaviors and cognitive states (Steinmetz et al., 2019). Experimentally, identifying these properties typically relies on probe localization through histology and expert annotation (Liu et al., 2021). However, these procedures are expensive, cannot be performed in vivo, and are subject to inter-rater variability (Carey et al., 2023), motivating the need for automated approaches that can infer neuronal identity directly from neural activity.

From a computational perspective, solving this task requires learning representations that capture biologically meaningful latent structure from neural signals alone. Prior work has shown that anatomical region and cell type can, to some extent, be decoded from activity patterns (Arora et al., 2026; Yu et al., 2025; Schneider et al., 2023a). However, the Brainwide Map dataset (IBL et al., 2025b) presents a substantially more challenging setting due to its broad anatomical coverage, large inter-animal variability, and diversity of recording conditions.

Table 1: **Task Suite 1: Behavior decoding performance.** Models are tuned, finetuned, and evaluated on a single session and single task at a time. We report Balanced Accuracy for sequence-level tasks and R^2 for frame-level tasks, except for Licks where we report D^2. All metrics are reported as the average over evaluation sessions ($\pm$ SEM over 5 seeds during evaluation). See Appendix E for details on metrics and ranking. Green shading denotes TSS, magenta denotes TSU. No shading denotes a single session baseline.

		Frame-level Tasks					Sequence-level Tasks			Avg
	Method	Licks (D^2)	Whisker (R^2)	Wheel (R^2)	RPaw (R^2)	LPaw (R^2)	Reward (Acc)	Choice (Acc)	Contrast (Acc)	Rank
Single-Session	Linear	0.151 ± 0.000	0.175 ± 0.000	0.219 ± 0.002	0.091 ± 0.000	0.084 ± 0.001	0.813 ± 0.004	0.614 ± 0.004	0.219 ± 0.001	7.73
	MLP	0.348 ± 0.001	0.268 ± 0.002	0.329 ± 0.001	0.157 ± 0.001	0.135 ± 0.001	0.877 ± 0.006	0.636 ± 0.004	0.224 ± 0.002	5.46
	GRU	0.510 ± 0.002	0.358 ± 0.002	0.300 ± 0.006	0.214 ± 0.002	0.176 ± 0.002	0.885 ± 0.005	0.644 ± 0.007	0.214 ± 0.001	3.96
	CNN	0.549 ± 0.008	0.389 ± 0.004	0.330 ± 0.004	0.225 ± 0.004	0.194 ± 0.002	0.887 ± 0.004	0.621 ± 0.002	0.205 ± 0.002	3.74
	CEBRA	0.327 ± 0.003	0.280 ± 0.003	0.200 ± 0.002	0.164 ± 0.004	0.121 ± 0.002	0.780 ± 0.008	0.521 ± 0.002	0.198 ± 0.001	7.44
	NDT	0.314 ± 0.001	0.303 ± 0.002	0.212 ± 0.003	0.163 ± 0.002	0.127 ± 0.001	0.871 ± 0.005	0.616 ± 0.003	0.208 ± 0.003	6.80
	POYO	0.559 ± 0.002	0.291 ± 0.004	0.300 ± 0.003	0.195 ± 0.002	0.136 ± 0.002	0.882 ± 0.002	0.630 ± 0.005	0.224 ± 0.002	4.80
Pretrain	POYO+	0.678 ± 0.003	0.380 ± 0.003	0.362 ± 0.001	0.249 ± 0.002	0.200 ± 0.000	0.898 ± 0.003	0.689 ± 0.003	0.267 ± 0.001	2.53
	POSSM	0.661 ± 0.002	0.351 ± 0.003	0.355 ± 0.002	0.237 ± 0.001	0.175 ± 0.002	0.887 ± 0.004	0.662 ± 0.005	0.220 ± 0.002	3.18
	NDT-Stitch	0.525 ± 0.002	0.384 ± 0.003	0.310 ± 0.002	0.219 ± 0.005	0.172 ± 0.003	0.863 ± 0.006	0.674 ± 0.001	0.229 ± 0.001	3.58
	MtM	0.469 ± 0.004	0.362 ± 0.004	0.323 ± 0.006	0.224 ± 0.003	0.167 ± 0.003	0.874 ± 0.003	0.646 ± 0.004	0.213 ± 0.002	4.05

Motivated by these challenges, TS3 evaluates whether models recover intrinsic and biologically grounded structure from neural activity. Because brain regions differ in their connectivity, functional roles, cell type composition, and population statistics, they induce consistent signatures in neural dynamics that persist across animals and experimental conditions (Tolossa et al., 2025; Schneider, 2025). We therefore evaluate brain region identification in a zero-shot transfer setting to test two complementary goals: whether learned representations uncover anatomical structure without explicit supervision, and whether they support practical in vivo localization in previously unseen animals.

TS3: Tasks. Given neural activity from individual units, the task is to predict the associated anatomical region label. We align all probes to the Allen Common Coordinate Framework (CCF) (Wang et al., 2020) and use Cosmos-level annotations from the IBL anatomical atlas (IBL et al., 2025b), resulting in a 10-class classification problem spanning coarse but functionally meaningful regions such as *Isocortex*, *Hippocampus*, and *Cerebellum*. We evaluate both the *single-unit* setting, where neurons are classified independently, and the *multi-unit* setting, where predictions are informed by neighboring units on the same probe. All evaluations are performed on held-out animals in a zero-shot setting, requiring generalization across recording sessions, probe placements, and inter-animal variability without using region labels from evaluation sessions during training (Appendix D.3).

TS3: Evaluation Setup. We consider two across-animal evaluation settings designed to probe different aspects of generalization. The first is the *Transductive Zero-Shot* setting, where models are adapted on held-out animals using some form of supervision external to the brain region labels (e.g., behavioral labels as in POYO+), allowing them to adjust representations to the new domain before evaluation. This setting reflects realistic scenarios where some data from a new subject is available for adaptation. The second is the *Inductive Zero-Shot* setting, where pretrained models are evaluated on entirely unseen animals and probe configurations without any adaptation, providing a stringent test of whether learned representations capture invariant features of neural identity. Performance is measured using macro-averaged F1 score to account for class imbalance.

TS3: Baselines. We evaluate methods for learning unit-level representations, including NEMO (Yu et al., 2025) and NuCLR (Arora et al., 2026), by pretraining them using their respective training objectives. Since these models are capable of generating embeddings for a new neural population via an inference step, they are representative baselines for the Inductive setting. In addition, we provide simple baselines for the inductive setting using interspike interval (ISI) and local field potential (LFP) features directly as embeddings (see Appendix B for details on the available LFP data). We also use LOLCAT from (Schneider et al., 2023a) which is directly trained on the brain region classification task, without any pretraining. Since LOLCAT extracts features directly using the brain region classification task for supervision, it represents a TSS baseline for TS3, whereas NEMO and NuCLR fall under TSU. Finally, we evaluate unit-level embeddings from models originally pretrained for TS1 (POYO+ (Azabou et al., 2025), POSSM (Ryoo et al., 2025)) and TS2 (NDT-Stitch (Ye et al., 2023; Zhang et al., 2024) and MtM (Zhang et al., 2024)) as representative models for the Transductive setting. These models are finetuned on eval sessions using their pretraining objectives, as they require gradients to generate embeddings on the new neural population (Appendix F).

Table 2: **Task Suite 2: Co-smoothing and forecasting performance.** Models are tuned, finetuned, and evaluated on a single session and single task at a time. We report D^2 and bps averaged over evaluation sessions ($\pm$ SEM over 5 seeds during evaluation). See Appendix E for details on metrics and ranking. Green shading denotes `TSS`, magenta denotes `TSU`. No shading denotes a single session baseline.

	Method	Co-smoothing		Forecasting		Avg
		D^2	bps	D^2	bps	Rank
SS	Autoencoder	0.088 ± 0.000	0.200 ± 0.001	0.037 ± 0.001	0.065 ± 0.001	4.81
SS	LFADS	0.176 ± 0.000	0.421 ± 0.001	0.144 ± 0.000	0.325 ± 0.001	2.34
SS	NDT	0.132 ± 0.001	0.304 ± 0.003	0.158 ± 0.000	0.343 ± 0.001	2.79
Pre	NDT-Stitch	0.157 ± 0.001	0.369 ± 0.002	0.177 ± 0.000	0.384 ± 0.000	1.64
Pre	MtM	0.191 ± 0.001	0.459 ± 0.002	0.131 ± 0.001	0.288 ± 0.001	2.57

Table 3: **Task Suite 3: Brain region prediction performance.** Results are reported as the macro-average F1 score ($\pm$ SEM over 5 seeds). Seeds parameterize embedding generation through target-session calibration (transductive) or pretraining/supervised training (inductive), except for the ISI/LFP baselines where they parameterize the MLP probe. All pretrained models are evaluated in zero-shot transfer w.r.t. region labels. Green shading denotes `TSS`, magenta denotes `TSU`, and no shading is used for untrained embeddings.

		Single Unit (F1)		Multi-Unit (F1)	
Regime	Method	Linear Probe	MLP Probe	Linear Probe	MLP Probe
Transductive	POYO+	0.099 ± 0.003	0.120 ± 0.005	0.072 ± 0.001	0.132 ± 0.008
Transductive	POSSM	0.103 ± 0.002	0.123 ± 0.003	0.104 ± 0.004	0.133 ± 0.006
Transductive	NDT-Stitch	0.136 ± 0.007	0.167 ± 0.007	0.155 ± 0.011	0.187 ± 0.008
Transductive	MtM	0.137 ± 0.004	0.140 ± 0.006	0.159 ± 0.009	0.158 ± 0.007
Inductive	ISI Baseline	0.263	0.380 ± 0.002	0.355	0.519 ± 0.006
Inductive	LFP Baseline	0.405	0.492 ± 0.005	0.403	0.496 ± 0.006
Inductive	LOLCAT	0.359 ± 0.003	–	0.473 ± 0.004	–
Inductive	NEMO	0.447 ± 0.003	0.471 ± 0.002	0.579 ± 0.007	0.605 ± 0.004
Inductive	NuCLR	0.616 ± 0.006	0.626 ± 0.005	0.654 ± 0.010	0.645 ± 0.003

4 Benchmark Results

TS1 Results. In Table 1, we compare behavior decoding performance across pretrained models on both frame-level and sequence-level tasks. Unsurprisingly, `TSS` models pretrained directly on the decoding tasks (green) generally outperform `TSU` models pretrained on a different objective (magenta). However, across-task transfer remains surprisingly effective on frame-level tasks such as whisker and right paw, and sequence-level tasks such as reward and choice decoding. This suggests that pretrained representations capture latent neural structure that generalizes beyond their original supervision.

We also compare with single-session models to benchmark performance against representative architectures, though these require extensive tuning. Notably, POYO and NDT are both improved by their pretrained counterparts: POYO improves in average rank over tasks from $4.80 \rightarrow 2.53$ with POYO+, and NDT improves from $6.80 \rightarrow 3.58$ with NDT-Stitch (ranking procedure detailed in Appendix E.4). Among the single-session baselines, recurrent and convolutional models achieve strong performance across most tasks, particularly for high-signal variables such as licking and whisker motion. Nonetheless, POYO+ achieves the best average rank among our baselines, and POSSM the second best. Together, these results indicate that pretraining yields representations that transfer well across animals and decoding objectives.

TS2 Results. In Table 2, we evaluate neural activity prediction using both co-smoothing and forecasting tasks. Across both metrics, the best-performing model on each task is pretrained, demonstrating that large-scale pretraining improves the ability to capture structured neural dynamics. In particular, NDT-Stitch achieves the strongest forecasting performance, while MtM performs best on co-smoothing, suggesting that different pretraining objectives emphasize complementary aspects of neural dynamics. Interestingly, NDT-Stitch is effective on forecasting despite being pretrained using a non-causal masked prediction objective, suggesting that representations learned through temporal reconstruction transfer well even with shift in the task distribution. Conversely, MtM performs particularly well on co-smoothing, likely due to the fact that for most training steps it employs a spatial masking strategy, hence a closer alignment to the downstream objective.

TS3 Results. In Table 3, we evaluate brain region identification across held-out animals in both transductive and inductive settings. Across all evaluation regimes, methods specifically designed to learn neuron-level representations substantially outperform models that optimize for a different objective, such as behavioral decoding or neural prediction. In particular, NuCLR achieves the strongest performance across both single-unit and multi-unit settings, reaching a macro-F1 of 0.654 in the multi-unit linear probe setting. NEMO also performs strongly with a peak macro-F1 of 0.605 with a multi-unit MLP probe. Both of these models focus on extracting invariant features of neuron identity. On the other hand, transductive methods such as POYO+, POSSM, NDT-Stitch, and MtM achieve substantially lower performance, suggesting that representations optimized for other objectives do not automatically organize around anatomical structure. Nevertheless, NDT-Stitch and MtM outperform POYO+ and POSSM, indicating that masked neural prediction may preserve more region-specific structure than behaviorally-aligned decoding objectives.

We also generally observe improvements from single-unit to multi-unit evaluation, suggesting that neighboring neurons contain complementary information that improves localization accuracy. Interestingly, LOLCAT underperforms some `TSU` baselines despite optimizing its latents for the downstream task. Overall, these results demonstrate that anatomical structure can emerge from neural activity representations in a zero-shot setting, while also highlighting the importance of pretraining objectives that explicitly encourage stable neuron-level representations across animals.

5 Discussion

This benchmark evaluates model generalization across three axes: behavior, dynamics, and anatomy, providing a unified view of how large-scale models pretrained on neural data transfer across animals and downstream tasks. Across these axes, we observe a consistent and encouraging trend: pretrained models can act as strong generalist representations that rapidly adapt to new animals and recording conditions. At the same time, performance differences across task suites reveal important distinctions in what current pretraining objectives capture most effectively and where gaps still remain.

Our results suggest that multi-session neural models may help amortize the substantial cost of training and tuning models for individual sessions and tasks. Achieving strong single-session performance often requires extensive hyperparameter optimization, whereas pretrained models can be adapted to new animals and tasks using a largely standardized fine-tuning setup (see Appendix H for a breakdown of compute requirements per model, where we see single-session tuning is substantially more compute-intensive). Across our experiments, pretrained models were finetuned using minimal task-specific tuning, yet achieved competitive or superior performance across all settings. This suggests that large-scale pretraining not only improves transfer, but may also reduce the engineering burden required to deploy neural models across diverse datasets and experimental conditions. Still, important gaps remain in the ability of current models to transfer across new sessions and neural populations. For example, pretrained behavioral decoding and neural dynamics models still require target-session calibration to learn parameters specific to the new recording population, such as unit and session embeddings or session-specific read-in/read-out stitchers. Developing models for these tasks that require minimal or zero-shot calibration is a natural direction for future work that `BrainWideBench` is well suited to evaluate.

These conclusions should be interpreted in light of several limitations of the current benchmark. While `BrainWideBench` tests generalization across animals, evaluation is limited to a single species (mouse), recording modality (Neuropixels electrophysiology), and behavioral paradigm (visually-guided decision-making). Furthermore, the held-out set was chosen to keep evaluation feasible while preserving broad anatomical coverage, but the result is a modest cohort of 13 animals. Thus, rankings are more reliable in aggregate than in fine-grained comparison: broad groupings of models hold up consistently across resampled splits, but rankings within a group are more sensitive to which animals are held out and should not be read as statistically distinguishable pointwise comparisons.

`BrainWideBench` provides a framework for systematically measuring progress toward general-purpose models of neural activity. By absorbing the substantial cost of baseline training, dataset curation and standardization into a single, fully documented evaluation suite, `BrainWideBench` allows researchers to focus on modeling and scientific questions, accelerating progress toward models that are genuinely useful for neuroscience.

AI USE STATEMENT

Generative AI tools were used to aid and polish the writing and to draft sections of the paper. The authors take responsibility for the final content of this work.

ETHICS STATEMENT

`BrainWideBench` reuses publicly released neural and behavioral recordings from mice in the International Brain Laboratory Brainwide Map dataset (IBL, 2022). The source study reports that its animal procedures were approved by the relevant institutional ethics committees (IBL et al., 2025b). This benchmark does not involve research with human subjects. Reusing existing recordings supports further analyses without requiring new animal recordings for the benchmark. The results are limited to the species, recording modality, and behavioral paradigm studied and should not be interpreted as evidence of clinical validity or direct generalization to humans.

REPRODUCIBILITY STATEMENT

We document data quality control and preprocessing in Appendices A and B, and the pretraining, evaluation, and within-session splits in Appendix C. Task definitions and evaluation metrics are specified in Appendices D and E. Model architectures, hyperparameter search spaces and training procedures, and computational resources are described in Appendices F, G, and H, respectively. Per-session results and additional evaluations are reported in Appendix I. The benchmark code is linked in the abstract. All pretrained model checkpoints and training logs will be publicly released upon publication; they are withheld from this submission to preserve anonymity.

REFERENCES

Larry F Abbott, Dora E Angelaki, Matteo Carandini, Anne K Churchland, Yang Dan, Peter Dayan, Sophie Deneve, Ila Fiete, Surya Ganguli, Kenneth D Harris, et al. An international laboratory for systems and computational neuroscience. *Neuron*, 96(6):1213–1218, 2017.

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Edgar D Adrian. The impulses produced by sensory nerve endings: Part i. *The Journal of physiology*, 61(1):49, 1926.

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 2623–2631, 2019.

Richard Antonello, Aditya Vaidya, and Alexander Huth. Scaling laws for language encoding models in fmri. *Advances in Neural Information Processing Systems*, 36:21895–21907, 2023.

Vinam Arora, Divyansha Lachi, Ian Jarratt Knight, Mehdi Azabou, Blake Aaron Richards, Cole Lincoln Hurwitz, Josh Siegle, and Eva L Dyer. Know thyself by knowing others: Learning neuron identity from population context. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=zt3RKc6VBp.

Mehdi Azabou, Vinam Arora, Venkataramana Ganesh, Ximeng Mao, Santosh Nachimuthu, Michael Mendelson, Blake Richards, Matthew Perich, Guillaume Lajoie, and Eva Dyer. A unified, scalable framework for neural population decoding. *Advances in Neural Information Processing Systems*, 36, 2024.

Mehdi Azabou, Krystal Xuejing Pan, Vinam Arora, Ian Jarratt Knight, Eva L Dyer, and Blake Aaron Richards. Multi-session, multi-task neural decoding from distinct cell-types and brain regions. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=IuU0wcO0mo.

Hubert Banville, Yohann Benchetrit, Stéphane d'Ascoli, Jérémy Rapin, and Jean-Rémi King. Scaling laws for decoding images from brain activity. *arXiv preprint arXiv:2501.15322*, 2025.

Maxime Beau, Federico D'Agostino, Ago Lajko, Gabriela Martínez, and Dimitar Kostadinov. Neuropyxels: loading, processing and plotting neuropixels data in python. *Zenodo https://doi.org/10.5281/ZENODO*, 5509776, 2021.

Maxime Beau, David J. Herzfeld, Francisco Naveros, Marie E. Hemelt, Federico D'Agostino, Marlies Oostland, Alvaro Sánchez-López, Young Yoon Chung, Michael Maibach, Hannah N. Stabb, et al. A deep-learning strategy to identify cell types across species from high-density extracellular recordings. *Cell*, 2025. doi: 10.1016/j.cell.2025.01.041.

Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994. doi: 10.1109/72.279181.

Philipp Berens, Jeremy Freeman, Thomas Deneux, Nikolay Chenkov, Thomas McColgan, Artur Speiser, Jakob H Macke, Srinivas C Turaga, Patrick Mineault, Peter Rupprecht, et al. Community-based benchmarking improves spike rate inference from two-photon calcium imaging data. *PLoS computational biology*, 14(5):e1006157, 2018.

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. *Advances in neural information processing systems*, 24, 2011.

Dan Biderman, Matthew R Whiteway, Cole Hurwitz, Nicholas Greenspan, Robert S Lee, Ankit Vishnubhotla, Richard Warren, Federico Pedraja, Dillon Noone, Michael M Schartner, et al. Lightning pose: improved animal pose estimation via semi-supervised learning, bayesian ensembling and cloud-native open-source tools. *Nature Methods*, 21(7):1316–1328, 2024.

Benjamin Blankertz, K-R Muller, Gabriel Curio, Theresa M Vaughan, Gerwin Schalk, Jonathan R Wolpaw, Alois Schlogl, Christa Neuper, Gert Pfurtscheller, Thilo Hinterberger, et al. The bci competition 2003: progress and perspectives in detection and discrimination of eeg single trials. *IEEE transactions on biomedical engineering*, 51(6):1044–1051, 2004.

Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

A Colin Cameron and Frank AG Windmeijer. An r-squared measure of goodness of fit for some common nonlinear regression models. *Journal of econometrics*, 77(2):329–342, 1997.

Harry Carey, Michael Pegios, Lewis Martin, Chris Saleeba, Anita J Turner, Nicholas A Everett, Ingvild E Bjerke, Maja A Puchades, Jan G Bjaalie, and Simon McMullan. Deepslice: rapid fully automatic registration of mouse brain imaging to a volumetric atlas. *Nature Communications*, 14 (1):5884, 2023.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations, 2020. URL https://arxiv.org/abs/2002.05709.

Mehdi Cherti, Romain Beaumont, Ross Wightman, Mitchell Wortsman, Gabriel Ilharco, Cade Gordon, Christoph Schuhmann, Ludwig Schmidt, and Jenia Jitsev. Reproducible scaling laws for contrastive language-image learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2818–2829, 2023.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans (eds.), *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1179. URL https://aclanthology.org/D14-1179/.

Mark M Churchland, John P Cunningham, Matthew T Kaufman, Justin D Foster, Paul Nuyujukian, Stephen I Ryu, and Krishna V Shenoy. Neural population dynamics during reaching. *Nature*, 487 (7405):51–56, 2012.

John P Cunningham and Byron M Yu. Dimensionality reduction for large-scale neural recordings. *Nature neuroscience*, 17(11):1500–1509, 2014.

Kabir V. Dabholkar and Omri Barak. When predict can also explain: Few-shot prediction to select better neural latents. *PLOS Computational Biology*, 21(12):1–25, 12 2026. doi: 10.1371/journal.pcbi.1013789. URL https://doi.org/10.1371/journal.pcbi.1013789.

Saskia EJ de Vries, Jerome A Lecoq, Michael A Buice, Peter A Groblewski, Gabriel K Ocker, Michael Oliver, David Feng, Nicholas Cain, Peter Ledochowitsch, Daniel Millman, et al. A large-scale standardized physiological survey reveals functional organization of the mouse visual cortex. *Nature neuroscience*, 23(1):138–151, 2020.

Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7(Jan):1–30, 2006.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pp. 4171–4186, 2019.

Alan D Dorval. Probability distributions of the logarithm of inter-spike intervals yield accurate entropy estimates from small datasets. *Journal of neuroscience methods*, 173(1):129–139, 2008.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Rotem Dror, Gili Baumer, Segev Shlomov, and Roi Reichart. The hitchhiker's guide to testing statistical significance in natural language processing. In *Proceedings of the 56th annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 1383–1392, 2018.

Yu Duan, Hamza Chaudhry, Misha B Ahrens, Christopher Harvey, Matthew G Perich, Karl Deisseroth, and Kanaka Rajan. Poco: Scalable neural forecasting through population conditioning. *Advances in neural information processing systems*, 38:23631–23669, 2026.

Eva Dyer and Blake Richards. Accepting "the bitter lesson" and embracing the brain's complexity. *SPECTRUM*, 2025.

Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

Charles Findling, Felix Hubert, IBL, Luigi Acerbi, Brandon Benson, Julius Benson, Daniel Birman, Niccolò Bonacchi, E Kelly Buchanan, Sebastian Bruijns, et al. Brain-wide representations of prior information in mouse decision-making. *Nature*, 645(8079):192–200, 2025.

Wulfram Gerstner, Werner M Kistler, Richard Naud, and Liam Paninski. *Neuronal dynamics: From single neurons to networks and models of cognition*. Cambridge University Press, 2014.

Joshua I Glaser, Ari S Benjamin, Raeed H Chowdhury, Matthew G Perich, Lee E Miller, and Konrad P Kording. Machine learning for neural decoding. *eneuro*, 7(4), 2020.

Evren Gokcen, Anna I Jasper, João D Semedo, Amin Zandvakili, Adam Kohn, Christian K Machens, and Byron M Yu. Disentangling the flow of signals between populations of neurons. *Nature Computational Science*, 2(8):512–525, 2022.

Carl Gold, Darrell A Henze, Christof Koch, and Gyorgy Buzsaki. On the origin of the extracellular action potential waveform: a modeling study. *Journal of neurophysiology*, 95(5):3113–3128, 2006.

Adam Gosztolai, Robert L Peach, Alexis Arnaudon, Mauricio Barahona, and Pierre Vandergheynst. Marble: interpretable representations of neural population dynamics using geometric deep learning. *Nature Methods*, 22(3):612–620, 2025.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=tEYskw1VY2.

Albert Gu, Karan Goel, and Christopher Re. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=uYLFoz1vlAC.

Kenneth D Harris. Nonsense correlations in neuroscience. *biorxiv*, pp. 2020–11, 2020.

Gal Haspel, Ben Baker, Isabel Beets, Edward S Boyden, Jeffrey Brown, George Church, Netta Cohen, Daniel Colon-Ramos, Eva Dyer, Christopher Fang-Yen, et al. The time is ripe to reverse engineer an entire nervous system: simulating behavior from neural interactions. *arXiv preprint arXiv:2308.06578*, 2023.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 16000–16009, 2022.

Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. Bag of tricks for image classification with convolutional neural networks. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 558–567. IEEE, 2019.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006. doi: 10.1126/science.1127647. URL https://www.science.org/doi/abs/10.1126/science.1127647.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8): 1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Thomas Hennigan, Eric Noland, Katherine Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karén Simonyan, Erich Elsen, Oriol Vinyals, Jack Rae, and Laurent Sifre. An empirical analysis of compute-optimal large language model training. In *Advances in Neural Information Processing Systems*, volume 35, pp. 30016–30030. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/c1e2faff6f588870935f114ebe04a3e5-Paper-Conference.pdf.

Xiao Shi Huang, Felipe Perez, Jimmy Ba, and Maksims Volkovs. Improving transformer optimization through better initialization. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 4475–4483. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/huang20f.html.

Cole Hurwitz, Nina Kudryashova, Arno Onken, and Matthias H Hennig. Building population models for large-scale neural recordings: Opportunities and pitfalls. *Current opinion in neurobiology*, 70: 64–73, 2021.

IBL. Data release - brainwide map - technical paper. *Figshare*, 2022. doi: 10.6084/m9.figshare.21400815.v7. URL https://figshare.com/articles/preprint/Data_release_-_Brainwide_map_-_Q4_2022/21400815?file=49286065.

IBL, Valeria Aguillon-Rodriguez, Dora Angelaki, Hannah Bayer, Niccolo Bonacchi, Matteo Carandini, Fanny Cazettes, Gaelle Chapuis, Anne K Churchland, Yang Dan, Eric Dewitt, Mayo Faulkner, Hamish Forrest, Laura Haetzel, Michael Häusser, Sonja B Hofer, Fei Hu, Anup Khanal, Christopher Krasniak, Ines Laranjeira, Zachary F Mainen, Guido Meijer, Nathaniel J Miska, Thomas D Mrsic-Flogel, Masayoshi Murakami, Jean-Paul Noel, Alejandro Pan-Vazquez, Cyrille Rossant, Joshua Sanders, Karolina Socha, Rebecca Terry, Anne E Urai, Hernando Vergara, Miles Wells,

Christian J Wilson, Ilana B Witten, Lauren E Wool, and Anthony M Zador. Standardized and reproducible measurement of decision-making in mice. *eLife*, 10:e63711, may 2021. ISSN 2050-084X. doi: 10.7554/eLife.63711. URL https://doi.org/10.7554/eLife.63711.

IBL, Dan Birman, Niccolo Bonacchi, Kelly Buchanan, Gaelle Chapuis, Julia Huntenburg, Guido Meijer, Liam Paninski, Michael Schartner, Karel Svoboda, Matthew Whiteway, Miles Wells, and Olivier Winter. Video hardware and software for the international brain laboratory. *figshare*, 2022a. doi: 10.6084/m9.figshare.19694452. URL https://doi.org/10.6084/m9.figshare.19694452.

IBL, Julien Boussard, Gaëlle A Chapuis, Kenneth D Harris, Christopher Langfield, Liam Paninski, Cyrille Rossant, Noam Roth, Nicholas A. Steinmetz, Charlie Windolf, and Olivier Winter. Spike sorting pipeline for the International Brain Laboratory. *figshare*, 5 2022b. doi: 10.6084/m9.figshare.19705522.v4. URL https://figshare.com/articles/online_resource/Spike_sorting_pipeline_for_the_International_Brain_Laboratory/19705522.

IBL, Niccolò Bonacchi, Gaelle A Chapuis, Anne K Churchland, Eric EJ DeWitt, Mayo Faulkner, Kenneth D Harris, Julia M Huntenburg, Max Hunter, Ines C Laranjeira, et al. A modular architecture for organizing, processing and sharing neurophysiology data. *Nature methods*, 20(3):403–407, 2023.

IBL, Kush Banga, Julius Benson, Jai Bhagat, Dan Biderman, Daniel Birman, Niccolò Bonacchi, Sebastian A Bruijns, Kelly Buchanan, Robert AA Campbell, et al. Reproducibility of in vivo electrophysiological measurements in mice. *Elife*, 13:RP100840, 2025a.

IBL, Brandon Benson, Julius Benson, Daniel Birman, Niccolò Bonacchi, Matteo Carandini, Joana A Catarino, Gaelle A Chapuis, Anne K Churchland, Yang Dan, Peter Dayan, Eric EJ DeWitt, Tatiana A Engel, Michele Fabbri, Mayo Faulkner, Ila Rani Fiete, Charles Findling, Laura Freitas-Silva, Berk Gerçek, Kenneth D Harris, Michael Häusser, Sonja B Hofer, Fei Hu, Félix Hubert, Julia M Huntenburg, Anup Khanal, Christopher Krasniak, Christopher Langdon, Petrina Y P Lau, Zachary F Mainen, Guido T Meijer, Nathaniel J Miska, Thomas D Mrsic-Flogel, Jean-Paul Noel, Kai Nylund, Alejandro Pan-Vazquez, Alexandre Pouget, Cyrille Rossant, Noam Roth, Rylan Schaeffer, Michael Schartner, Yanliang Shi, Karolina Z Socha, Nicholas A Steinmetz, Karel Svoboda, Anne E Urai, Miles J Wells, Steven Jon West, Matthew R Whiteway, Olivier Winter, and Ilana B Witten. A brain-wide map of neural activity during complex behaviour. *Nature*, 645(8079):177–191, 2025b. URL https://www.nature.com/articles/s41586-025-09235-0.

Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, et al. Perceiver io: A general architecture for structured inputs & outputs. *arXiv preprint arXiv:2107.14795*, 2021.

Xianyan Jia, Shutao Song, Wei He, Yangzihao Wang, Haidong Rong, Feihu Zhou, Liqiang Xie, Zhenyu Guo, Yuanzhou Yang, Liwei Yu, et al. Highly scalable deep learning training system with mixed-precision: Training imagenet in four minutes. *arXiv preprint arXiv:1807.11205*, 2018.

Linxing Preston Jiang, Shirui Chen, Emmanuel Tanumihardja, Xiaochuang Han, Weijia Shi, Eric Shea-Brown, and Rajesh PN Rao. Data heterogeneity limits the scaling effect of pretraining neural data transformers. *bioRxiv*, pp. 2025–05, 2025.

Yiqi Jiang, Kaiwen Sheng, Yujia Gao, Estefany Kelly Buchanan, Yu Shikano, Seung Je Woo, Yixiu Zhao, Tony Hyun Kim, Fatih Dinc, Scott Linderman, et al. Extracting task-relevant preserved dynamics from contrastive aligned neural recordings. *Advances in Neural Information Processing Systems*, 38:160999–161031, 2026.

James J Jun, Nicholas A Steinmetz, Joshua H Siegle, Daniel J Denman, Marius Bauza, Brian Barbarits, Albert K Lee, Costas A Anastassiou, Alexandru Andrei, Çağatay Aydın, et al. Fully integrated silicon probes for high-density recording of neural activity. *Nature*, 551(7679):232–236, 2017.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Jaivardhan Kapoor, Auguste Schulz, Julius Vetter, Felix C Pei, Richard Gao, and Jakob H. Macke. Latent diffusion for neural spiking data. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=ZX6CEo1Wtv.

Brianna M. Karpowicz, Joel Ye, Chaofei Fan, Pablo Tostado-Marcos, Fabio Rizzoglio, Clayton B Washington, Thiago Scodeler, Diogo S de Lucena, Samuel R. Nason-Tomaszewski, Matthew Mender, Xuan Ma, Ezequiel Matias Arneodo, Leigh Hochberg, Cynthia Chestek, Jaimie M. Henderson, Timothy Q Gentner, Vikash Gilja, Lee E. Miller, Adam G. Rouse, Robert Gaunt, Jennifer L Collinger, and Chethan Pandarinath. Few-shot algorithms for consistent neural decoding (FALCON) benchmark. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=FN02v4nD8y.

Brianna M Karpowicz, Yahia H Ali, Lahiru N Wimalasena, Andrew R Sedler, Mohammad Reza Keshtkaran, Kevin Bodkin, Xuan Ma, Daniel B Rubin, Ziv M Williams, Sydney S Cash, et al. Stabilizing brain-computer interfaces through alignment of latent dynamics. *Nature Communications*, 16(1):4662, 2025.

Mohammad Reza Keshtkaran, Andrew R Sedler, Raeed H Chowdhury, Raghav Tandon, Diya Basrai, Sarah L Nguyen, Hansem Sohn, Mehrdad Jazayeri, Lee E Miller, and Chethan Pandarinath. A large-scale neural network training framework for generalized estimation of single-trial population dynamics. *Nature methods*, 19(12):1572–1577, 2022.

Trung Le and Eli Shlizerman. Stndt: Modeling neural population activity with spatiotemporal transformers. *Advances in Neural Information Processing Systems*, 35:17926–17939, 2022.

Colin Lea, Rene Vidal, Austin Reiter, and Gregory D Hager. Temporal convolutional networks: A unified approach to action segmentation. In *European conference on computer vision*, pp. 47–54. Springer, 2016.

Liu D Liu, Susu Chen, Han Hou, Steven J West, Mayo Faulkner, International Brain Laboratory, Michael N Economo, Nuo Li, and Karel Svoboda. Accurate localization of linear probe electrode arrays across multiple brains. *ENeuro*, 8(6):ENEURO–0241, 2021.

Zicheng Liu, Jiahui Li, Siyuan Li, Zelin Zang, Cheng Tan, Yufei Huang, Yajing Bai, and Stan Z Li. Genbench: A benchmarking suite for systematic evaluation of genomic foundation models. *arXiv preprint arXiv:2406.01627*, 2024.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. URL https://arxiv.org/abs/1711.05101.

Jakob H Macke, Lars Buesing, John Cunningham, Byron M Yu, Krishna V Shenoy, and Maneesh Sahani. Empirical models of spiking in neural populations. In J. Shawe-Taylor, R. Zemel, P. Bartlett, F. Pereira, and K. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011. URL https://proceedings.neurips.cc/paper_files/paper/2011/file/7143d7fbadfa4693b9eec507d9d37443-Paper.pdf.

Jeremy Magland, James J Jun, Elizabeth Lovero, Alexander J Morley, Cole Lincoln Hurwitz, Alessio Paolo Buccino, Samuel Garcia, and Alex H Barnett. Spikeforest, reproducible web-facing ground-truth validation of automated neural spike sorters. *Elife*, 9:e55167, 2020.

Ximeng Mao, Nanda H. Krishna, Avery Hee-Woon Ryoo, Matthew G. Perich, and Guillaume Lajoie. Leveraging unlabelled data for generalizable neural population decoding, 2026. URL https://arxiv.org/abs/2607.14086.

Jonathan D. McCart, Andrew R. Sedler, Christopher Versteeg, Domenick Mifsud, Mattia Rigotti-Thompson, and Chethan Pandarinath. Diffusion-based generation of neural activity from disentangled latent codes, 2024. URL https://arxiv.org/abs/2407.21195.

Peter McCullagh, John A Nelder, and P McCullagh. *Generalized linear models*, volume 2. Chapman and hall London, 1989.

Yuki Minai, Joana Soldado-Magraner, Byron M. Yu, and Matthew A. Smith. OMiSO: Adaptive optimization of state-dependent brain stimulation to shape neural population states. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=OiqQGjNvyJ.

Pascal Notin, Aaron Kollasch, Daniel Ritter, Lood Van Niekerk, Steffanie Paul, Han Spinner, Nathan Rollins, Ada Shaw, Rose Orenbuch, Ruben Weitzman, et al. Proteingym: Large-scale benchmarks for protein fitness prediction and design. *Advances in Neural Information Processing Systems*, 36: 64331–64379, 2023.

Chethan Pandarinath, Paul Nuyujukian, Christine H Blabe, Brittany L Sorice, Jad Saab, Francis R Willett, Leigh R Hochberg, Krishna V Shenoy, and Jaimie M Henderson. High performance communication by people with paralysis using an intracortical brain-computer interface. *elife*, 6: e18554, 2017.

Chethan Pandarinath, K Cora Ames, Abigail A Russo, Ali Farshchian, Lee E Miller, Eva L Dyer, and Jonathan C Kao. Latent factors and dynamics in motor cortex and their application to brain–machine interfaces. *Journal of Neuroscience*, 38(44):9390–9401, 2018a.

Chethan Pandarinath, Daniel J O'Shea, Jasmine Collins, Rafal Jozefowicz, Sergey D Stavisky, Jonathan C Kao, Eric M Trautmann, Matthew T Kaufman, Stephen I Ryu, Leigh R Hochberg, et al. Inferring single-trial neural population dynamics using sequential auto-encoders. *Nature methods*, 15(10):805–815, 2018b.

Liam Paninski, Jonathan Pillow, and Jeremy Lewi. Statistical models for neural encoding, decoding, and optimal stimulus design. *Progress in brain research*, 165:493–507, 2007.

Stefano Panzeri, Nicolas Brunel, Nikos K Logothetis, and Christoph Kayser. Sensory neural codes using multiplexed temporal scales. *Trends in neurosciences*, 33(3):111–120, 2010.

Felix C Pei, Joel Ye, David M. Zoltowski, Anqi Wu, Raeed Hasan Chowdhury, Hansem Sohn, Joseph E O'Doherty, Krishna V. Shenoy, Matthew Kaufman, Mark M Churchland, Mehrdad Jazayeri, Lee E. Miller, Jonathan W. Pillow, Il Memming Park, Eva L Dyer, and Chethan Pandarinath. Neural latents benchmark '21: Evaluating latent variable models of neural population activity. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. URL https://openreview.net/forum?id=KVMS3fl4Rsv.

Donald H Perkel, George L Gerstein, and George P Moore. Neuronal spike trains and stochastic point processes: I. the single spike train. *Biophysical journal*, 7(4):391–418, 1967.

Hans-Peter Piepho. An algorithm for a letter-based representation of all-pairwise comparisons. *Journal of Computational and Graphical Statistics*, 13(2):456–466, 2004.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.

Yuchen Ren, Zhiyuan Chen, Lifeng Qiao, Hongtai Jing, Yuchen Cai, Sheng Xu, Peng Ye, Xinzhu Ma, Siqi Sun, Hongliang Yan, et al. Beacon: Benchmark for comprehensive rna tasks and language models. *Advances in Neural Information Processing Systems*, 37:92891–92921, 2024.

Avery Hee-Woon Ryoo, Nanda H Krishna, Ximeng Mao, Mehdi Azabou, Eva L Dyer, Matthew G Perich, and Guillaume Lajoie. Generalizable, real-time neural decoding with hybrid state-space models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=1i4wNFgHDd.

Mostafa Safaie, Joanna C Chang, Junchol Park, Lee E Miller, Joshua T Dudman, Matthew G Perich, and Juan A Gallego. Preserved neural dynamics across animals performing similar behaviour. *Nature*, 623(7988):765–771, 2023.

Motoshige Sato, Kenichi Tomeoka, Ilya Horiguchi, Kai Arulkumaran, Ryota Kanai, and Shuntaro Sasai. Scaling law in neural data: Non-invasive speech decoding with 175 hours of eeg data. *arXiv preprint arXiv:2407.07595*, 2024.

Aidan Schneider. *Robust Embeddings of Genetics, Anatomy, and State Decoded from a Neuron's Activity*. PhD thesis, Washington University in St. Louis, 2025.

Aidan Schneider, Mehdi Azabou, Louis McDougall-Vigier, David F Parks, Sahara Ensley, Kiran Bhaskaran-Nair, Tomasz Nowakowski, Eva L Dyer, and Keith B Hengen. Transcriptomic cell type structures in vivo neuronal activity across multiple timescales. *Cell reports*, 42(4), 2023a.

Steffen Schneider, Jin Hwa Lee, and Mackenzie Weygandt Mathis. Learnable latent embeddings for joint behavioural and neural analysis. *Nature*, May 2023b. ISSN 1476-4687. doi: 10.1038/s41586-023-06031-6. URL https://doi.org/10.1038/s41586-023-06031-6.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th annual meeting of the association for computational linguistics (volume 1: long papers)*, pp. 1715–1725, 2016.

Qi She and Anqi Wu. Neural dynamics discovery via gaussian process recurrent neural networks. In Ryan P. Adams and Vibhav Gogate (eds.), *Proceedings of The 35th Uncertainty in Artificial Intelligence Conference*, volume 115 of *Proceedings of Machine Learning Research*, pp. 454–464. PMLR, 22–25 Jul 2020. URL https://proceedings.mlr.press/v115/she20a.html.

Leslie N. Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using large learning rates, 2018. URL https://arxiv.org/abs/1708.07120.

Nicholas A Steinmetz, Peter Zatka-Haas, Matteo Carandini, and Kenneth D Harris. Distributed coding of choice, action and engagement across the mouse brain. *Nature*, 576(7786):266–273, 2019.

Nicholas A Steinmetz, Cagatay Aydin, Anna Lebedeva, Michael Okun, Marius Pachitariu, Marius Bauza, Maxime Beau, Jai Bhagat, Claudia Böhm, Martijn Broux, et al. Neuropixels 2.0: A miniaturized high-density probe for stable, long-term brain recordings. *Science*, 372(6539): eabf4588, 2021.

Carsen Stringer and Marius Pachitariu. Analysis methods for large-scale neuronal recordings. *Science*, 386(6722):eadp7429, 2024.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Simon Thorpe and Jacques Gautrais. Rank order coding. In *Computational Neuroscience: Trends in Research, 1998*, pp. 113–118. Springer, 1998.

Gemechu Bekele Tolossa, Aidan M Schneider, Eva Dyer, and Keith B Hengen. Neurons throughout the brain embed robust signatures of their anatomical location into spike trains. *Elife*, 13:RP101506, 2025.

Polina Turishcheva, Paul G. Fahey, Michaela Vystrčilová, Laura Hansel, Rachel Froebe, Kayla Ponder, Yongrong Qiu, Konstantin F. Willeke, Mohammad Bashiri, Ruslan Baikulov, Yu Zhu, Lei Ma, Shan Yu, Tiejun Huang, Bryan M. Li, Wolf De Wulf, Nina Kudryashova, Matthias H. Hennig, Nathalie L. Rochefort, Arno Onken, Eric Wang, Zhiwei Ding, Andreas S. Tolias, Fabian H. Sinz, and Alexander S. Ecker. Retrospective for the dynamic sensorium competition for predicting large-scale mouse primary visual cortex activity from videos. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Pauli Virtanen, Ralf Gommers, Travis E Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, 17(3):261–272, 2020.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in neural information processing systems*, 32, 2019.

Eric Y Wang, Paul G Fahey, Zhuokun Ding, Stelios Papadopoulos, Kayla Ponder, Marissa A Weis, Andersen Chang, Taliah Muhammad, Saumil Patel, Zhiwei Ding, et al. Foundation model of neural activity predicts response to new stimulus types. *Nature*, 640(8058):470–477, 2025.

Quanxin Wang, Song-Lin Ding, Yang Li, Josh Royall, David Feng, Phil Lesnar, Nile Graddis, Maitham Naeemi, Benjamin Facer, Anh Ho, et al. The allen mouse brain common coordinate framework: a 3d reference atlas. *Cell*, 181(4):936–953, 2020.

Xiaoxi Wei, A Aldo Faisal, Moritz Grosse-Wentrup, Alexandre Gramfort, Sylvain Chevallier, Vinay Jayaram, Camille Jeunet, Stylianos Bakas, Siegfried Ludwig, Konstantinos Barmpas, et al. 2021 beetl competition: Advancing transfer learning for subject independence and heterogenous eeg data sets. In *NeurIPS 2021 Competitions and Demonstrations Track*, pp. 205–219. PMLR, 2022.

Konstantin F Willeke, Paul G Fahey, Mohammad Bashiri, Laura Pede, Max F Burg, Christoph Blessing, Santiago A Cadena, Zhiwei Ding, Konstantin-Klemens Lurz, Kayla Ponder, et al. The sensorium competition on predicting large-scale mouse primary visual cortex activity. *arXiv preprint arXiv:2206.08666*, 2022.

Konstantin Friedrich Willeke, Polina Turishcheva, Alex Gilbert, Goirik Chakrabarty, Hasan Atakan Bedel, Paul G. Fahey, Yongrong Qiu, Marissa A. Weis, Michaela Vystrčilová, Taliah Muhammad, Lydia Ntanavara, Rachel E Froebe, Kayla Ponder, Zheng Huan Tan, Emin Orhan, Erick Cobos, Sophia Sanborn, Katrin Franke, Fabian H. Sinz, Alexander S. Ecker, and Andreas S. Tolias. Omnimouse: Scaling properties of multi-modal, multi-task brain models on 150b neural tokens. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=mEw4lhAn0F.

Francis R Willett, Jingyuan Li, Trung Le, Chaofei Fan, Mingfei Chen, Eli Shlizerman, Yue Chen, Xin Zheng, Tatsuo S Okubo, Tyler Benster, et al. Brain-to-text benchmark'24: Lessons learned. *arXiv preprint arXiv:2412.17227*, 2024.

Ji Xia, Yizi Zhang, Shuqi Wang, Genevera I. Allen, Liam Paninski, Cole Lincoln Hurwitz, and Kenneth D. Miller. Inpainting the neural picture: Inferring unrecorded brain area dynamics from multi-animal datasets. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=YufSVJxDgt.

Joel Ye and Chethan Pandarinath. Representation learning for neural population activity with neural data transformers. *arXiv preprint arXiv:2108.01210*, 2021.

Joel Ye, Jennifer Collinger, Leila Wehbe, and Robert Gaunt. Neural data transformer 2: multi-context pretraining for neural spiking activity. *Advances in Neural Information Processing Systems*, 36: 80352–80374, 2023.

Joel Ye, Fabio Rizzoglio, Adam Smoulder, Hongwei Mao, Xuan Ma, Patrick Marino, Raeed Chowdhury, Dalton Moore, Gary Blumenthal, William Hockeimer, et al. A generalist intracortical motor decoder. *bioRxiv*, 2025.

Byron M Yu, John Cunningham, Gopal Santhanam, Stephen Ryu, Krishna V Shenoy, and Maneesh Sahani. Gaussian-process factor analysis for low-dimensional single-trial analysis of neural population activity. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou (eds.), *Advances in Neural Information Processing Systems*, volume 21. Curran Associates, Inc., 2008. URL https://proceedings.neurips.cc/paper_files/paper/2008/file/ad972f10e0800b49d76fed33a21f6698-Paper.pdf.

Han Yu, Hanrui Lyu, YiXun Xu, Charlie Windolf, Eric Kenji Lee, Fan Yang, Andrew M Shelton, Olivier Winter, International Brain Laboratory, Eva L Dyer, Chandramouli Chandrasekaran, Nicholas A. Steinmetz, Liam Paninski, and Cole Lincoln Hurwitz. In vivo cell-type and brain region classification via multimodal contrastive learning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=10JOlFIPjt.

Yizi Zhang, Yanchen Wang, Donato M Jiménez-Benetó, Zixuan Wang, Mehdi Azabou, Blake Richards, Renee Tung, Olivier Winter, Eva Dyer, Liam Paninski, et al. Towards a "universal translator" for neural dynamics at single-cell, single-spike resolution. *Advances in Neural Information Processing Systems*, 37:80495–80521, 2024.

Yizi Zhang, Yanchen Wang, Mehdi Azabou, Alexandre Andre, Zixuan Wang, Hanrui Lyu, Eva L Dyer, Liam Paninski, Cole Lincoln Hurwitz, et al. Neural encoding and decoding at scale. In *Forty-second International Conference on Machine Learning*, 2025.

Supplementary Material

`BrainWideBench`: Benchmarking large-scale pretraining and across-animal transfer in multi-region neural recordings.

CONTENTS

A Data Quality Control **21**

B Data Processing **24**

C Data Splits **27**

D Task Suites **33**

E Metrics and Ranking **40**

F Overview of Baseline Models **42**

G Hyperparameter Selection **52**

H Computing Resources **77**

I Additional Results **78**

J Related Work **91**

A Data Quality Control

Large-scale neurophysiology datasets of the scale and complexity of the IBL Brainwide Map (BWM) inevitably contain data of variable quality across sessions and modalities. Recording artifacts, hardware failures, signal extraction issues (e.g., spike sorting and pose estimation), and imperfect behavioral engagement can all degrade data quality and vary in severity across sessions. Rather than applying a single binary data quality filter to each session, we characterize quality along multiple independent axes: session-level behavior; neural activity, including raw electrophysiology, spike sorting, probe alignment and individual unit quality; video-based behavior signals; and trial-level behavior. We make all quality metrics available as part of the benchmark release, enabling downstream users to apply more or less stringent filters as appropriate for their use case. We describe each quality metric below, then detail how these metrics are combined to construct the pretraining and eval splits in Appendix C.

A.1 Session-level Quality Control

Sessions were included in the BWM data release (IBL, 2022) if the mouse performed at least 250 trials, with performance of at least 90% correct on 100%-contrast trials for both left and right stimulus blocks. Sessions were additionally required to pass a collection of hardware diagnostic tests, whose precise definitions are available at `https://int-brain-lab.github.io/iblenv/_autosummary/ibllib.qc.task_metrics.html`. These criteria constitute a baseline behavioral inclusion criterion applied uniformly across the full dataset prior to any further quality assessment, and therefore all sessions in the benchmark dataset pass these controls.

A.2 Neural Quality Control

Probe Alignment (`qc_neural_alignment`). The alignment quality was determined by running a brain-region decoder algorithm (not yet published), which assigns a brain region label to each channel of a probe insertion according to electrophysiology signatures (e.g., power in the Local Field Potential). By comparing the output of this decoder and the brain region label originally annotated by experimenters, the severity of the discrepancy was labeled as `HIGH`, `LOW` or `NONE`. `HIGH` severity alignments are marked as `FAIL`; `LOW` severity alignments are marked as `WARNING`; and alignments with `NONE` severity are marked `PASS`.

Alignment quality was not used to determine the pretraining set, but it did shape the evaluation set: a subject was eligible for evaluation only if each of its sessions contained at least one probe with both `qc_overall` and `qc_neural_alignment` marked `PASS`, and evaluation sessions retaining no such probe were discarded. Because insertions are extracted per session rather than individually, the released evaluation build still contains 4 of 45 insertions whose alignment is marked `FAIL`. These are excluded at scoring time: Task Suite 3 (brain region identification), where accurate anatomical labeling of units is essential, scores only units on probes whose alignment is marked `PASS`. This excludes both `FAIL` and `WARNING` alignments, the latter affecting only pretraining sessions, since no evaluation probe carries a `WARNING` alignment.

Raw Electrophysiology (`qc_neural_raw`). Probes were manually marked as `PASS` / `WARNING` / `FAIL` depending on whether images presenting short snippets of raw destriped action potential (AP) data presented noise or artifacts, as done previously for the Brainwide Map data release (IBL et al., 2022b).

Spike Sorting (`qc_neural_spikesorting`). Probes were manually marked as `PASS` / `WARNING` / `FAIL` depending on whether the spike raster presented discontinuity, jitter, horizontal band and loss of spikes, as done previously for the Brainwide Map data release (IBL et al., 2022b). Insertions presenting such issues in a very mild manner, or not at all, were marked as `PASS`.

Individual Units. Units produced by the spike sorting pipeline (IBL et al., 2022b) were assessed for quality using three criteria established in prior work (IBL et al., 2025a): (i) median spike waveform amplitude greater than 50 μV; (ii) noise cut-off below 5 (a.u.); and (iii) absence of refractory period violations. Units passing all three criteria are designated as "good" units. Unit-level quality is assessed for every session regardless of the session-level raw electrophysiology or spike sorting

outcome. All units, "good" and otherwise, are released as part of the benchmark. Good-unit filtering is applied during evaluation of Task Suites 2 and 3 (neural activity prediction and brain region identification), where unit quality directly affects the validity of the neural signal. Task Suite 1 (behavioral decoding) is evaluated on all units, as decoder trained on population activity can in principle learn to down-weight uninformative units. Furthermore, all units with a firing rate less than 1 Hz are excluded from evaluation in Task Suites 2 and 3. See Table A1 for unit counts with varying filters applied.

Overall Neural QC (`qc_neural`). A probe-level neural QC label `qc_neural` is computed as the minimum (most severe) label across `qc_neural_raw` and `qc_neural_spikesorting`. Probe alignment quality is tracked separately and used only for Task Suite 3 exclusions as described above.

Table A1: Neural Unit Attrition Across Filtering Stages

Filtering Stage	# Sessions	Units	Unit-hours	% Retained
Original Brain Wide Map	459	621,733	—	—
Unfiltered `BrainWideBench`	452	610,701	830,324	100%
+ Probe QC	452	601,124	816,371	98.3%
+ Firing Rate	452	446,311	606,728	73.1%
+ Unit QC	452	67,160	90,390	11.0%

Note: Unfiltered `BrainWideBench` corresponds to the same data as the Original Brain Wide Map, except for the exclusion of 7 sessions that failed QC on the evaluation animals.

A.3 Behavioral Quality Control

Sessions were recorded from three video cameras: two side cameras ("left" and "right") that capture orthogonal views of the face and upper body of the mouse, and a "body" camera that captures the back of the mouse. We only utilize the "left" video in the benchmark, acquired at 60 frames per second. The IBL video processing pipeline (IBL et al., 2022a) produces two major outputs relevant to this benchmark: pose estimates from Lightning Pose for the left and right paws and the tongue (Biderman et al., 2024), and an estimate of the motion of the whisker pad computed as the mean absolute pixel difference between successive frames in a region of interest (ROI) around the whisker pad. Three video-derived behavioral signals are subject to additional manual quality assessment, each assigned a `PASS`, `WARNING`, or `FAIL` label.

Paw Poses (`qc_behavior_paws`). Paw pose estimation is assigned `FAIL` if obvious timestamp errors are present, as evidenced by a noisy peri-event time histogram (PETH), and `PASS` otherwise (Fig. A1A).

Lick Rate (`qc_behavior_licks`). Lick rate is computed from the tongue pose estimates (IBL et al., 2022a). Lick rate is assigned `FAIL` if timestamp issues are apparent in other Lightning Pose traces for either camera, if no discernible increase in lick rate is observed at reward time, or if one ROI around the mouth is incorrect without a compensating strong oscillatory signal during reward consumption. `WARNING` is assigned if an increase around reward time is present but weak, if both ROIs are acceptable but the oscillatory pattern is moderate, or if the oscillatory pattern changes substantially across the session in a manner that could reflect either a real behavioral change or a tracking artifact. `PASS` requires strong oscillatory licking signal with both cameras yielding acceptable ROIs (Fig. A1B).

Whisker Pad Motion Energy (`qc_behavior_motion_energy`). Whisker pad motion energy is assigned `FAIL` if the ROI is not properly centered on the whisker pad or if timestamp errors produce a noisy PETH. `WARNING` is assigned if the stimulus-aligned PETH is present but weak. `PASS` is assigned otherwise (Fig. A1C).

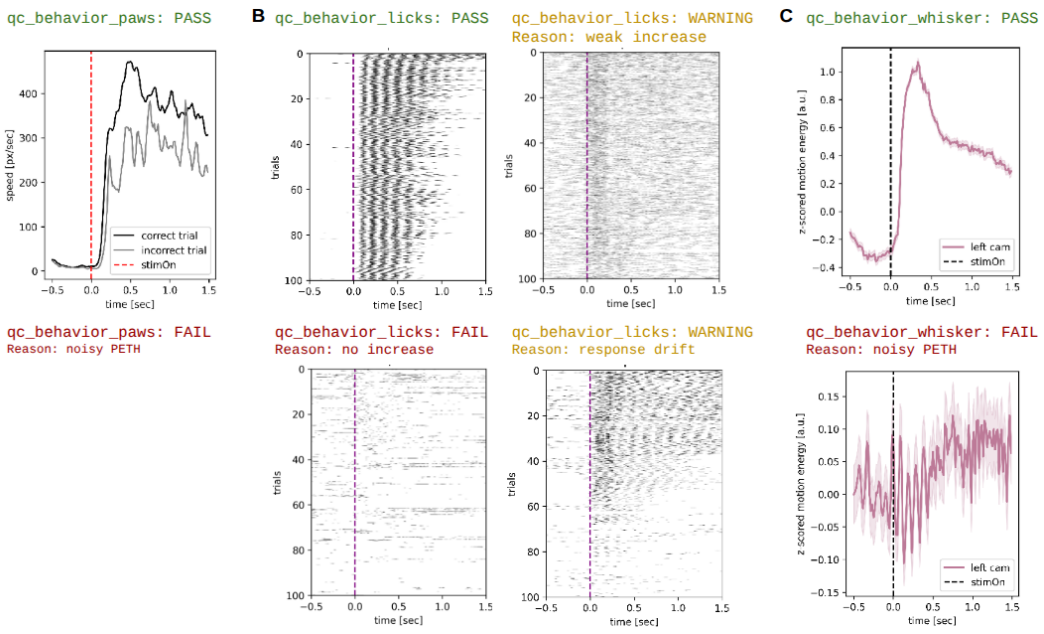


Figure A1: **Behavior QC. A**: *Top*: Example session with `qc_behavior_paws=PASS`, showing paw trajectories averaged separately for correct and incorrect trials, aligned to stimulus onset. No sessions had `WARNING` or `FAIL` labels for paws. **B**: *Top left*: Example session with `qc_behavior_licks=PASS`, showing lick times for correct trials aligned to feedback onset. *Bottom left*: Example session with `qc_behavior_licks=FAIL`, where there is no discernible increase in lick rate during the reward period, likely due to poor tongue tracking. *Right*: Two example sessions with `qc_behavior_licks=WARNING`, showing weak but present lick rate increase (*Top*) and non-stationary lick rate pattern over the course of the session (*Bottom*). **C**: *Top*: Example session with `qc_behavior_motion_energy=PASS`, showing whisker pad motion energy averaged over correct trials, aligned to stimulus onset. *Bottom*: Example session with `qc_behavior_motion_energy=FAIL`, where the signal is dominated by noise. No sessions were marked `WARNING`.

Overall Behavioral QC (`qc_behavior`). A session-level behavioral QC label `qc_behavior` is computed as the minimum (most severe) label across `qc_behavior_paws`, `qc_behavior_licks` and `qc_behavior_motion_energy`.

A.4 Trial-level Quality Control

Within individual sessions, several filters were applied to remove trials with missing data or unusual behavior. Trials were excluded if one of the following trial events could not be detected: `stimOn_times`, `probabilityLeft`, `firstMovement_times`, `choice`, `feedback_times`, and `feedbackType`. Trials were further excluded if the time between stimulus onset and the first movement of the wheel (a measure of reaction time) was longer than 10.0 s, which eliminates trials where the animal may not be actively engaged in the task.

B DATA PROCESSING

Raw data were accessed via the International Brain Laboratory's Brainwide Map data release (IBL, 2022) using the ONE API under the MIT license (https://docs.internationalbrainlab.org/notebooks_external/2025_data_release_brainwidemap.html). All processed outputs are serialized using the `brainsets` format, a structured HDF5-based schema designed for neural population data that supports arbitrary temporal resolution and heterogeneous modalities within a unified session object (https://brainsets.readthedocs.io). Each session is represented as a `Data` object whose attributes expose the processed neural and behavioral streams described below; for instance, the unified neural population activity is accessible as `data.spikes`.

Neural Activity (`spikes`). Spiking activity and cluster metadata were extracted for all available Neuropixels probe insertions within a session. To account for multiple probes, probe-local cluster IDs were offset to generate globally unique unit indices across the entire session. Anatomical coordinates for each unit were mapped to the Beryl brain region atlas. Finally, spike timestamps, amplitudes, depths, and unique unit indices from all probes were concatenated and temporally sorted into a unified, irregular time series representing the session-wide population activity. Note, the irregular time series object enables the data to be stored unbinned, allowing each benchmark user to bin the neural activity to their desired bin size. The unified neural population is exposed as `data.spikes`.

Continuous Behavioral Processing and Alignment Methodology. All continuous behavioral traces were extracted, aligned to a common temporal grid, and optionally normalized (see below). Behavioral time series were aligned to a uniform 50 Hz sampling rate, accommodating missing frames or gaps using a 10 ms gap tolerance during alignment (see below).

Signal Resampling and Anti-Aliasing Filtering. To prevent aliasing artifacts during the downsampling and resampling of high-frequency continuous data (e.g., 1 kHz wheel kinematics or >60 Hz video tracking), an anti-aliasing low-pass filter was applied prior to rate reduction. Specifically, signals were processed using an 8th-order zero-phase filter (Butterworth filters provide better qualitative final results compared to Chebyshev) to attenuate high-frequency noise without introducing phase shifts into the behavioral time series. To maintain a conservative safety margin, the filter's cutoff frequency was strictly set to 80% of the target Nyquist frequency. This ensured high-fidelity signal representation when decimating the data onto the final 50 Hz temporal grid.

Wheel Kinematics (`wheel_speed`). Wheel position was monitored with dedicated rotary encoder hardware. Wheel angular velocity traces were extracted from position traces and aligned to the common timebase. To preserve high-frequency fidelity during alignment, the wheel data was first regularized to a 1 kHz sampling grid, strictly phase-aligned to the start and end timestamps of the video data. The 1 kHz trace was then sequentially decimated (downsampled) by factors of 10 and 2 to reach the target 50 Hz rate. Finally, wheel speed was derived by taking the absolute value of the downsampled velocity trace, yielding a non-negative scalar in physical (angular) units. The processed wheel signals are exposed as `data.wheel`.

Pose Dynamics (`left_paw_speed` and `right_paw_speed`). Pose tracking data, generated via Lightning Pose (Biderman et al., 2024), was extracted for the left and right paws from the left camera video feed, which captures the face and upper trunk of the mouse. Because the pose dataset does not inherently contain timestamps, it inherited the timestamp array from the corresponding left camera video.

Keypoint Kinematics. For each keypoint $k \in \{\text{left paw}, \text{right paw}\}$ with 2D position $\mathbf{p}_k(t) = (x_k(t), y_k(t))$, the instantaneous 2D velocity was computed by central-difference differentiation along the time axis, $\mathbf{v}_k(t) = d\mathbf{p}_k/dt$, and the scalar speed was obtained as its Euclidean magnitude, $s_k(t) = \|\mathbf{v}_k(t)\|_2$. The paw kinematics and their per-frame confidence flags are exposed as `data.paws`.

Per-Frame Confidence Masks. Lightning Pose returns, for each keypoint and each frame, a continuous *likelihood* score in $[0, 1]$ reflecting the network's confidence in the localization. We threshold these per-paw likelihoods at $\tau = 0.9$ to obtain a binary, per-frame reliability flag. Because the velocity (and hence speed) at frame t is computed via central differences on the regularized position trace, each

kinematic sample depends on its two temporal neighbors $t \pm 1$; a single low-confidence frame therefore contaminates the velocity estimates of its neighbors. To account for this, the binary likelihood mask is morphologically eroded by one sample in time, yielding the final per-paw confidence masks (`is_left_paw_confident`, `is_right_paw_confident`). These masks are propagated alongside the kinematic signals on the 50 Hz grid and are used downstream both to gate normalization statistics for the paw signals (see **Behavioral Normalization**) and to decode unreliable frames in TS1 evaluation.

Whisker Motion Energy (`whisker_motion_energy`). Whisker motion energy was extracted from the left camera video feed. The motion energy is a single scalar per frame, computed by taking the sum of the absolute value of pixel-level differences across consecutive frames (IBL et al., 2022a). Because native video sampling rates varied across sessions ($\sim$60 Hz or $\sim$150 Hz), the time series was first regularized onto a uniform timestamp grid and then resampled to the 50 Hz target rate. The whisker motion energy trace is exposed as `data.whisker`.

Licking Behavior (`licking_rate`). Licking events were recorded as discrete timestamps, precomputed on the IBL public database using pose tracking outputs on the tongue (IBL et al., 2022a). To align this point-process data with the continuous behavioral features, lick timestamps were binned directly onto the regularized 50 Hz timestamp grid established by the video data. The resulting binned counts were multiplied by the target sampling rate to yield a continuous lick rate (in Hz). The continuous lick rate is exposed as `data.licks`.

Behavioral Normalization. Normalization is applied to the pose- and video-derived signals (`left_paw_speed`, `right_paw_speed`, and `whisker_motion_energy`), which are expressed in raw pixel units. Their absolute scale is camera- and session-dependent and can take very large numerical values that would otherwise dominate the loss and gradient dynamics across modalities; *z-scoring* places these heterogeneously scaled signals on a common, well-conditioned numerical footing. By contrast, `wheel_speed` and `licking_rate` are *not* normalized: `wheel_speed` is already in physical units (rad/s) from the rotary encoder, and `licking_rate` is scale-controlled by construction, obtained by binning lick events on the 50 Hz grid and multiplying by the sampling rate to yield a count-process rate in Hz. Both are consumed in their native units.

Per-signal *z-scoring* was computed as $\tilde{x}(t) = (x(t) - \mu)/\max(\sigma, \varepsilon)$, with $\varepsilon = 10^{-8}$ guarding against zero-variance signals. To prevent leakage and to ensure that the normalization scale is representative of the data actually scored at evaluation, the statistics (μ, σ) are accumulated over the same task-aligned movement intervals that define the evaluation windows, restricted to the training split only. In other words, (μ, σ) are estimated on the train-split portion of the exact temporal windows on which the regression metrics are computed at test time, rather than on the full continuous session. For the paw kinematics, only frames passing the per-paw confidence mask (`is_left_paw_confident`, `is_right_paw_confident`) contribute to (μ, σ). This prevents low-confidence tracking artifacts from inflating the estimated scale and distorting the standardized signal.

Discrete Behavioral Labels. Three behavioral variables are extracted as trial-level targets for classification: stimulus contrast, choice, and reward. They are taken directly from the dataset, kept unchanged, and exposed under `data.task_aligned_intervals` as `.stimulus_contrast`, `.choice`, and `.reward`.

Dataset Builds. The dataset is released in two builds that differ only in the unit population they store; the sessions, the behavioral streams, and the temporal splits are identical across both. `all_units` applies no unit filtering. `selected_units` applies three filters before the spike data is written to disk: `firing_rate` (mean firing rate above 1.0 Hz), `unit_qc` (IBL unit QC label equal to 1.0, i.e. well-isolated units), and `probe_qc` (probe insertions carrying a passing QC label, applied to evaluation sessions only). Because the filters run at processing time, they are baked into the HDF5 files and cannot be changed without reprocessing. Which build a suite uses is dictated by what it predicts. TS2 and TS3 attach their targets to individual units: TS2 reconstructs a held-out neuron's activity and TS3 classifies a unit's brain region, so a poorly isolated unit is a corrupted target or a corrupted label rather than merely a noisy input, and both suites require `selected_units`. TS1 decodes behavior from the whole population, where each unit is only an input feature and filtering

Table A2: Composition of the two dataset builds. Sessions, subjects, and recording duration are identical across builds; only the stored unit population differs. Probes counts insertions that still contribute at least one unit. Duration is total session recording time.

Build	Regime	Sessions	Subjects	Probes	Units	Spikes	Duration (h)
`all_units`	pretrain	423	126	643	567,514	19.21 B	561.5
`all_units`	eval	29	13	45	43,187	1.47 B	40.7
`selected_units`	pretrain	423	126	642	63,382	3.75 B	561.5
`selected_units`	eval	29	13	37	3,778	0.21 B	40.7

discards usable signal; all reported TS1 baselines therefore use `all_units`. Pretraining imposes no requirement and may use either build. Table A2 reports the composition of each build.

Local Field Potential (LFP). In addition to spiking activity, Neuropixels probes simultaneously record the local field potential (LFP), a lower-frequency signal that reflects summed synaptic and oscillatory activity in the tissue surrounding each recording site. The LFP offers a complementary view of neural activity, and we include it both as an additional input modality for behavior decoding and anatomical localization, and as a resource to encourage the development of multimodal approaches.

The LFP can be used either in its raw form (e.g., with lfp2vec) or in a compressed representation. The benchmark uses the compressed LFP released through `lfpack`. The lfpack release covers the Brain Wide Map insertions in a single archive. `BrainWideBench` re-ships the 688 insertions inside its session cohort as one file per probe insertion. In this release, the signals are cleaned, downsampled, and compressed for efficient storage. The compression preserves the main LFP frequency bands but attenuates higher-frequency content, and periods of signal saturation are flagged and removed. Because analyses of the Brain Wide Map have so far relied primarily on spiking activity, methods for processing the LFP in this dataset are likely to continue evolving. In particular, more complete quality-control labels for the LFP are not yet available, and adding them would be an important extension of the benchmark.

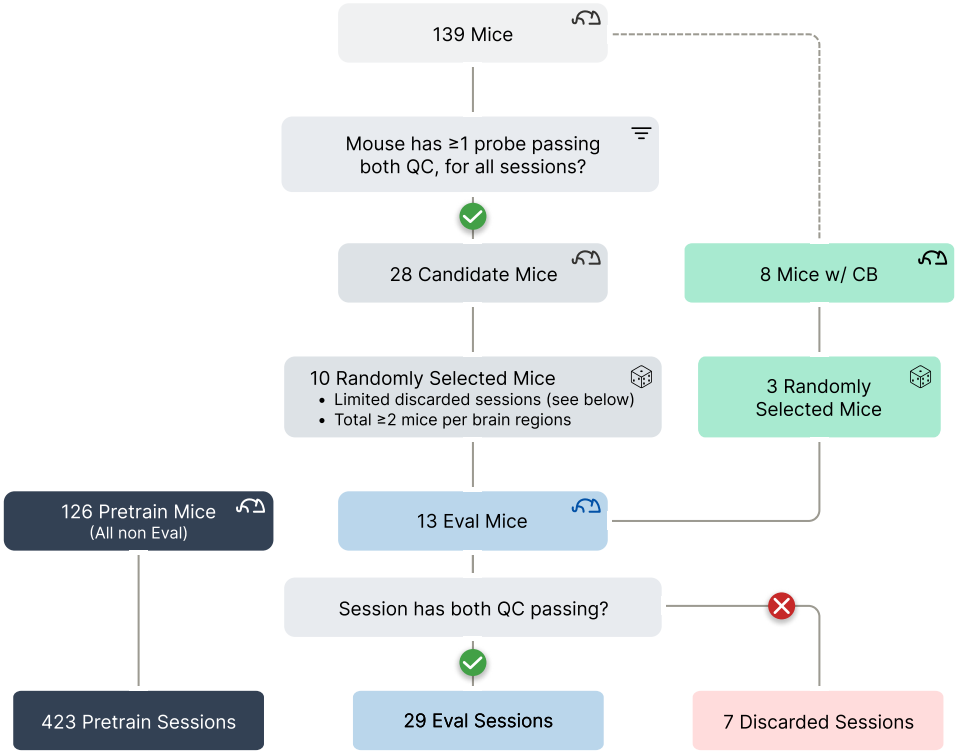


Figure A2: **Pretrain/Evaluation Mice Split.** Selection pipeline for the 13 evaluation mice from the full cohort of 139 mice. Starting from all 139 mice, 28 candidates are identified for whom every session has at least one probe passing both neural and alignment quality control and full behavioral quality control. From this pool, 10 mice are selected via a seeded greedy search optimizing for broad brain region coverage and minimal within-cohort session discards. Separately, 3 mice with confirmed cerebellum coverage are drawn from a curated pool of 8 cerebellum candidates, yielding 13 evaluation mice in total. All remaining 126 mice are assigned to pretraining. Within the evaluation cohort, 7 individual sessions failing quality control are discarded, resulting in 29 evaluation sessions and 423 pretraining sessions.

C Data Splits

C.1 Pretrain and Evaluation Split

The overarching design principle of the `BrainWideBench` pretraining/evaluation split is strict animal-level separation: no animal contributes data to both the pretraining and evaluation sets. This ensures that all three task suites evaluate genuine cross-animal generalization rather than within-animal memorization.

Two main criteria were considered to ensure effective transfer. First, the neural and behavioral data in the evaluation set must meet quality standards (passing the quality control considerations detailed in Appendix A). Second, the selected animals must provide broad coverage of brain regions, particularly for TS3 (Neuron Identity Prediction), where representative sampling across areas is essential for a meaningful evaluation of generalization. The overall selection process is illustrated in Figure A2 and described in detail in the following paragraphs.

Following the paper submission, we intend to host a submission-based leaderboard to accelerate community adoption of `BrainWideBench` and to maintain a living record of the state of the art. We note that `BrainWideBench` does not rely on a separate held-out evaluation set. Instead, all pretraining and evaluation sessions are openly released. This decision is deliberate. The benchmark is built on the Brainwide Map dataset, which is already publicly available, making any withheld sessions or signals straightforward to identify. Furthermore, the task suites impose heterogeneous requirements on the data: for example, withholding behavioral signals to prevent leakage in TS1 (Neural Decoding) would preclude their use as supervision for constructing neural embeddings in

TS3 (Neuron Identity Prediction). We therefore release all data in full and trust the community to engage with the benchmark in the spirit in which it is intended.

Identifying the evaluation candidate pool with Quality Control. We first identify 28 candidate mice by applying a two-stage quality filter. A session is considered usable if it passes behavioral quality control (`qc_behavior = PASS`) and contains at least one probe passing both neural quality control (`qc_neural = PASS`) and neural alignment quality control (`qc_neural_alignment = PASS`). We retain only mice for whom *every* recording session satisfies these criteria, yielding 28 animal-level candidates. This guarantee ensures that no general evaluation candidate contributes unusable sessions to the evaluation set. All remaining mice are assigned to the pretraining set (except for a cohort of mice with cerebellum recordings; see below).

Selecting evaluation mice via greedy search. From the 28 candidates, we apply a seeded greedy search to select 10 evaluation mice, jointly optimizing two criteria: (1) each selected mouse must contribute probe recordings spanning at least two distinct Cosmos-level brain regions, ensuring meaningful anatomical coverage per held-out subject; and (2) the number of probes discarded due to failing quality control within the evaluation cohort is minimized. Because cerebellum recordings are rare and scientifically valuable, 3 additional mice are sampled separately from a curated list of 8 cerebellum candidates. These cerebellum mice are not required to satisfy the animal-level criterion above; instead, sessions are filtered post-hoc, retaining only those that pass behavioral quality control and contain at least one probe passing both neural quality control and alignment quality control. This post-hoc filtering accounts for the 7 discarded sessions that differentiate the `BrainWideBench`'s 452 sessions from the original Brainwide Map's 459 sessions. The resulting evaluation set comprises 13 held-out mice and 29 sessions, with all remaining 126 mice forming the pretraining set of 423 sessions.

The resulting pretrain/evaluation split yields a brain region coverage distribution that is shown in Figure A3. Despite the strict animal-level separation and quality-driven selection, the evaluation set maintains broad anatomical coverage across all major brain regions with the number of mice, sessions, and probes per region remaining proportionally consistent between the two sets, in both the `all_units` and `selected_units` builds (Appendix B). Figure A5 illustrates the spatial distribution of individual neurons across the pretrain and evaluation mice.

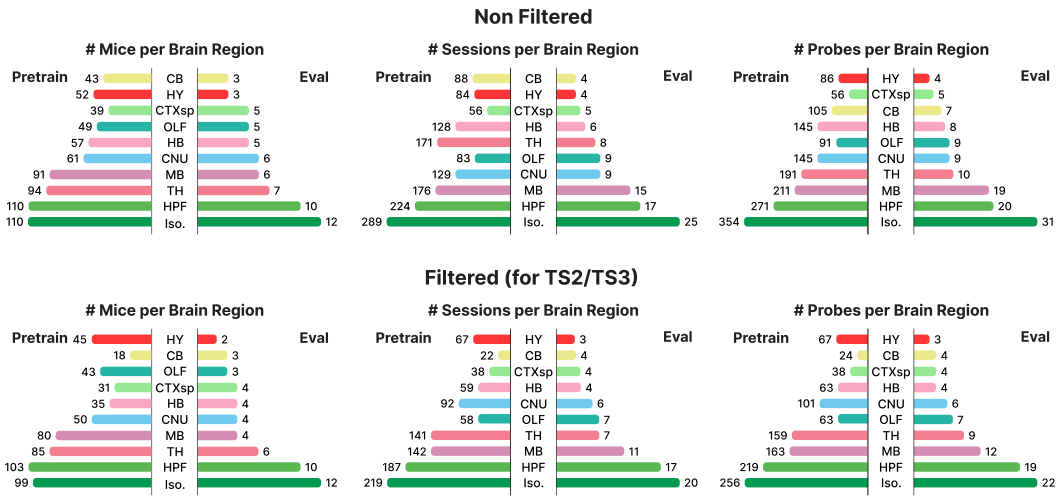


Figure A3: **Brain Region Distributions between Pretrain and Evaluation Sets.** Bar plots show the number of mice, sessions, and probes per Cosmos-level brain region, split by pretrain versus evaluation set. The top row shows the `all_units` build used for Task Suite 1, while the bottom row shows the `selected_units` build used for Task Suites 2 and 3.

Pretraining Scale. Table A3 reports the size of the pretraining corpus under three tokenization schemes: spike tokens, used by event-based models such as POYO; unit-bin tokens, used by patch-based models such as NDT2; and time-step tokens, used by models that keep units in the channel

Table A3: Pretraining corpus size over the full spike domain of all 423 sessions. Spike tokens count one token per spike; unit-bin tokens count one per (unit, 20 ms bin); time-step tokens count one per 20 ms bin with units in the channel dimension. Binned counts tile the domain with non-overlapping 1 s windows at 20 ms resolution.

				Tokens		
Build	Sessions	Units	Duration (h)	Spike	Unit-bin	Time-step
`all_units`	423	567,514	561.5	19.21 B	138.42 B	101,082,400
`selected_units`	423	63,382	561.5	3.75 B	15.36 B	101,080,400

dimension, such as NDT-Stitch. Counts cover the full spike domain of every pretraining session, with no restriction to task-aligned intervals, and assume 1 s windows and 20 ms bins. They are given for both builds; time-step tokens are near-identical across builds because the unit dimension never enters the token axis.

Sensitivity to the evaluation split Because model rankings are computed on a fixed set of held-out animals, we assessed whether our conclusions depend on the particular choice of evaluation split. We drew $B = 10{,}000$ bootstrap replicates by resampling the 13 held-out animals with replacement. Each drawn animal contributed all of its sessions, so that the resampling respects the hierarchical structure of the data (sessions nested within animals). For each replicate, we recomputed the average rank of every model across the tasks of TS1 (11 models, 8 tasks) and TS2 (5 models, 2 tasks), and we report the mean and the 95% percentile confidence interval of the resulting distribution (Fig. A4).

In TS1, POYO+ obtains the best average rank (2.53, 95% CI $[1.92, 3.22]$), followed by POSSM (3.19, $[2.71, 3.57]$) and NDT-Stitch (3.58, $[3.18, 4.05]$). The intervals of the six best-ranked models (POYO+, POSSM, NDT-Stitch, CNN, GRU, and MtM) overlap substantially, indicating that their fine-grained ordering depends on which animals are held out. Notably, the three best-ranked models are all fine-tuned multi-session models. The coarse structure of the ranking is robust to the split: POYO and MLP form an intermediate tier, and NDT, CEBRA, and Linear are consistently ranked lowest, with intervals that do not overlap those of the top group.In TS2, NDT-Stitch is ranked first (1.72, $[1.57, 1.87]$) with an interval that does not overlap any other model, and AE is consistently ranked last (4.86, $[4.73, 4.96]$). LFADS and MtM have overlapping intervals and cannot be reliably ordered relative to each other, whereas both are separated from NDT. These results confirm that the main conclusions drawn from our benchmark do not depend on the specific held-out animals.

C.2 Within-session Splits

Following the pretraining/evaluation split at the subject level, each session is further partitioned into training, validation, and test subsets at the trial level and specifically for each task suite as shown in Figure A6.

Pretraining sessions. For pretraining sessions, the first 90% of trials are assigned to the training split and the remaining 10% to the validation split. No test split is defined for pretraining sessions. This split is applied by default and can be updated to accommodate more advanced strategies, such as curriculum learning.

Evaluation sessions. For evaluation sessions, trials are divided into training (40%), validation (20%), and test (40%) splits. The relatively small training fraction favors models capable of learning from limited data, while the large test fraction maximizes the number of trials available for evaluation.

Task Suite 1. This setting utilizes a causal (non-interleaved) split: trials are assigned to splits in temporal order, such that all training trials precede all validation trials, which in turn precede all test trials. This design prevents decoding models from exploiting long-term signal drift, rather than learning the underlying structure of the data (Harris, 2020). Placing the validation split between training and test maximizes the temporal buffer between the two, further limiting the ability of models to exploit session-level drift. To enable longer context windows extending beyond 2 seconds (see Figure A11), a 20-second gap is enforced between adjacent splits.

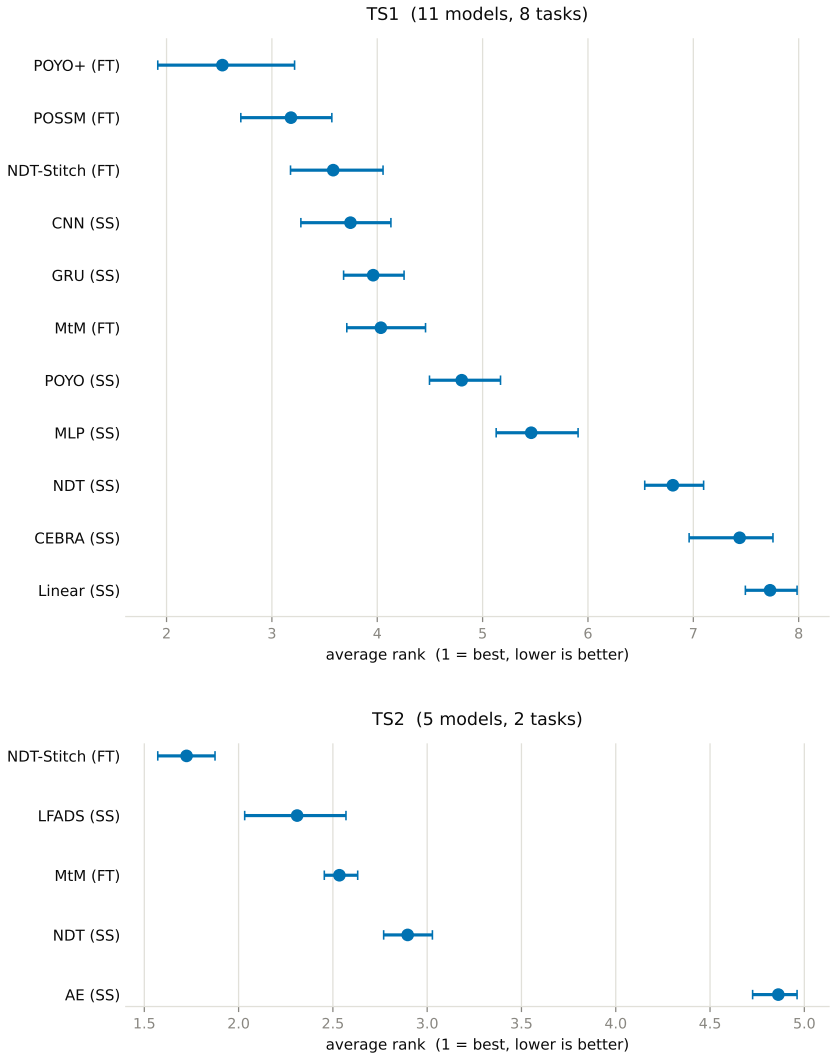


Figure A4: **Sensitivity of model rankings to the evaluation split.** Average rank (1 = best) of each model on TS1 (top; 11 models, 8 tasks) and TS2 (bottom; 5 models, 2 tasks), computed over $B = 10{,}000$ bootstrap replicates in which the 13 held-out animals were resampled with replacement, each contributing all of its sessions. Points show the bootstrap mean and error bars show 95% percentile confidence intervals. FT: fine-tuned multi-session model; SS: single-session model.

Task Suite 2. This setting utilizes an interleaved split of 5-minute temporal blocks following the pattern `train` `test` `train` `val` `test`, with a 20-second gap between blocks. This block structure reduces autocorrelation leakage that would arise from fully randomized trial-level splits while ensuring that each subset is large enough to contain representative samples of neural dynamics. Due to the strong non-stationary drift of neural recordings (IBL et al., 2022b), we found that replicating the causal split of TS1 led to a sharp drop in performance between validation and test sets (see Figure A7). As TS2 probes dynamical structure of neural populations, models trained on this task are more susceptible to non-stationary dynamics.

To motivate this choice, we compare TS2 task performance under two split strategies: causal (TS1) and interleaved (TS2) (Figure A7). Under the causal split, models are trained on the earlier portions of a recording and evaluated on the final held-out portion (Figure A6). We use the best-performing TS2 model (finetuned NDT-Stitch) across both tasks (forecasting and co-smoothing) and both metrics (D^2 and bps). We observe performance drops from validation to test under this strategy. This likely reflects non-stationarity in neural recordings (IBL et al., 2022b): as the temporal gap between



Figure A5: **Probe locations across 139 subjects in `BrainWideBench`.** Each brain shows the histology-aligned location of neurons in one mouse subject's brain across all acquisitions performed with that subject. Number of probe insertions ranges from 1 to 16 across subjects. Probe locations include added jitter (mean 0 s.d. 300 μm Gaussian) for better visualization.

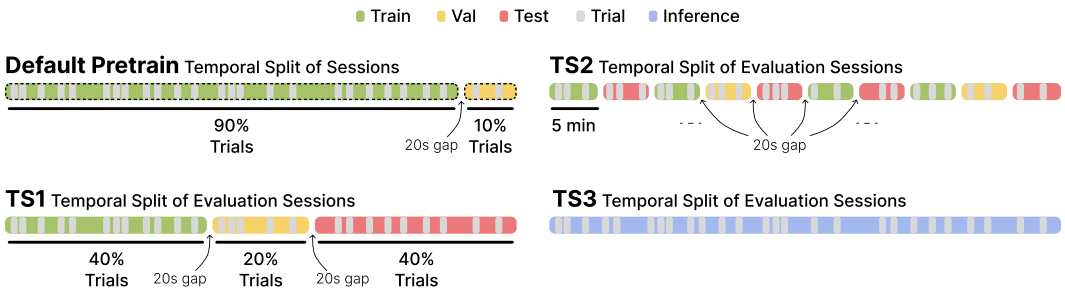


Figure A6: **Train/Validation/Test Splits Across Pretraining and Evaluation Sessions.** Each bar represents a single session, with colored segments indicating the temporal assignment of trials to training (green), validation (yellow), test (red) and inference (blue) splits. Splits are separated by 20-second gaps to prevent leakage and to accommodate longer context windows. Trial windows are shown in gray. Pretraining sessions use a 90/10 train/val split with no test set. Each evaluation task suite uses a split configuration tailored to its objectives: TS1 uses a causal 40/20/40 train/val/test split; TS2 uses interleaved 5-minute blocks; and TS3 uses the full session for inference.

train and test grows, probe drift and spike sorting artifacts cause neural dynamics to shift, making prediction of a fixed set of neurons increasingly ill-defined. Interleaved splits mitigate this by mixing train, validation, and test windows throughout the session, yielding more stable performance across splits.

We additionally examine how the choice of block duration affects test performance (Figure A8). Performance varies monotonically with block size: longer blocks reduce autocorrelation leakage but increase the difficulty of generalization across splits, while shorter blocks have the opposite effect. We select a block duration of 5 minutes (300 s) as a compromise between the two tasks: the geometric

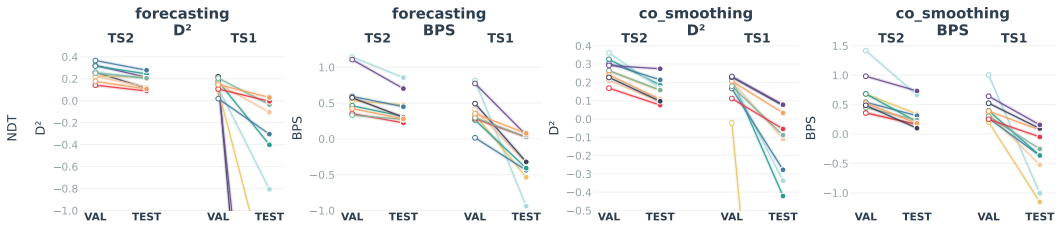


Figure A7: **TS2 Performance Under Causal (TS1) versus Interleaved (TS2) Split Strategies.** Each panel shows validation and test performance for a single-session NDT-Stitch model finetuned on 10 randomly selected evaluation sessions, across four task and metric combinations: forecasting (D^2), forecasting (BPS), co-smoothing (D^2), and co-smoothing (BPS). Each line connects the validation and test performance of a single session, with the left half of each panel showing the interleaved (TS2) split and the right half showing the causal (TS1) split. Under the interleaved split, validation and test performance are broadly consistent, reflecting the stationarity of the estimation problem. Under the causal split, performance drops sharply from validation to test across nearly all sessions and metrics, illustrating the detrimental effect of neural non-stationarity over long timescales. The slight reduction in validation performance between the TS2 and TS1 regimes is attributable to hyperparameters having been tuned for the interleaved split setting.

elbow occurs at 300 s for co-smoothing and 390 s for forecasting, and performance decreases only marginally between these two values.

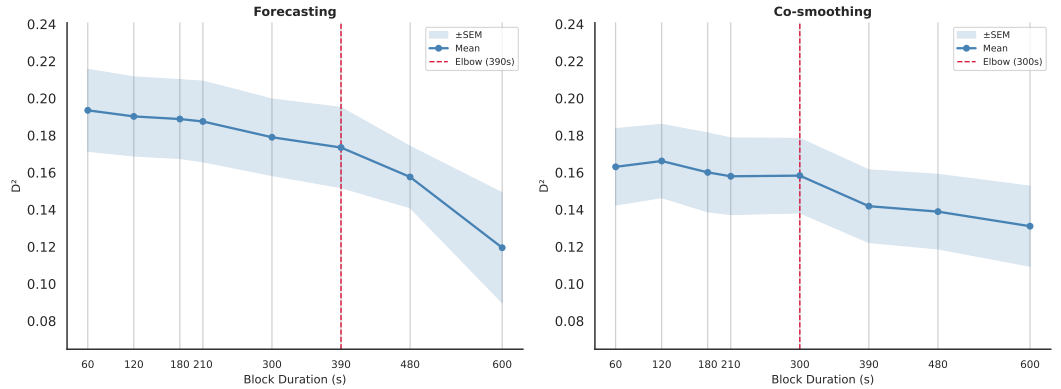


Figure A8: **Effect of Block Duration on Test D^2 Score.** Mean test D^2 ($\pm$ SEM) across 10 fine-tuned NDT models as a function of block duration, shown for the Forecasting (left) and Co-smoothing (right) tasks. Block duration is the length, in seconds, of the contiguous recording segments used to define the non-causal train/val/test splits for TS2. The dashed line marks the geometric elbow, defined as the block duration whose mean D^2 is at maximum perpendicular distance from the line connecting the first and last points.

Task Suite 3. This setting involves classifying the brain region of each recorded unit in a zero-shot setting and, by nature, does not depend on any particular temporal splitting. In the inductive setting, a model may use all available session data at test time (inference only, no gradient updates) to predict unit labels. For transductive models that first calibrate on a downstream task (e.g., behavior prediction in TS1 or neural activity prediction in TS2), the standard convention applies: the training split is used for gradient updates and the validation split for early stopping. The result is a set of embeddings for the new neural population that was learned on the evaluation data using a training objective external to the brain region task. Note that in our baseline evaluation, models operating in the transductive setting have access to less session data for embedding generation, since only the training split contributed to updating the unit embeddings and validation was used to select a set of embeddings. Nonetheless, nothing prevents such models from calibrating on the full session prior to transfer, but doing so requires an additional run.

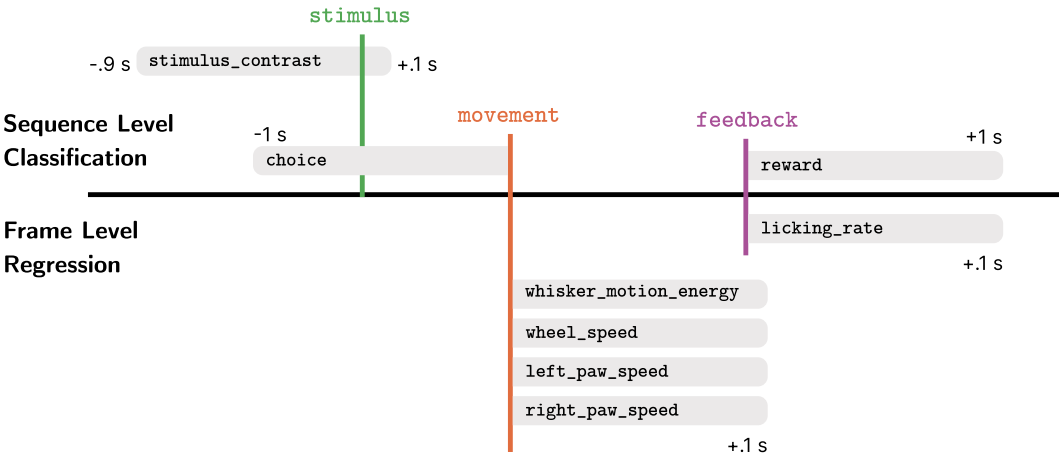


Figure A9: **Task-Aligned Decoding Windows for Behavior Prediction (TS1).** Illustration of the time windows used for each behavioral decoding target, aligned to three key trial events: stimulus onset (green), movement onset (orange), and feedback onset (purple). Sequence-level classification targets (top) include `stimulus_contrast`, decoded from a window around stimulus onset, and `choice` and `reward`, decoded from windows ending at and starting from movement onset and feedback onset, respectively. Frame-level regression targets (bottom) include `whisker_motion_energy`, `wheel_speed`, `left_paw_speed`, and `right_paw_speed`, decoded from a 1 second window starting at movement onset, and `licking_rate`, decoded from a 1 second window starting at feedback onset. Windows are designed to avoid leakage between task phases while capturing the most task-relevant period of neural and behavioral activity for each target.

D Task Suites

In this section, we provide details on the three task suites incorporated into our benchmark.

D.1 TS1 Details: Behavioral Decoding

Task Structure Overview. We define 8 decoding tasks in total: 5 at the frame level and 3 at the sequence level (see Table A4). To minimize confounds and maximize the ability to decode intent, we standardize all target windows to a fixed duration of 1 second. An overview of these windows is provided in Figure A9.

Table A4: Summary of decoding tasks with associated intervals, class counts, and evaluation metrics for TS1 (Behavior Prediction). Note: D^2 is defined in Appendix E.

Level	Variable Name	Interval Start	Interval End	Num Classes	Metric
Frame	whisker_motion_energy	movement_onset_time	movement_onset_time + 1.0	–	R^2, Pearson's r
	wheel_speed	movement_onset_time	movement_onset_time + 1.0	–	R^2, Pearson's r
	left_paw_speed	movement_onset_time	movement_onset_time + 1.0	–	R^2, Pearson's r
	right_paw_speed	movement_onset_time	movement_onset_time + 1.0	–	R^2, Pearson's r
	licking_rate	feedback_time	feedback_time + 1.0	–	D^2
Seq.	stimulus_contrast	stim_on_time − 0.9	stim_on_time + 0.1	5	**Balanced Acc**, $F1$ Score, AP
	choice	movement_onset_time	movement_onset_time + 1.0	2	**Balanced Acc**, $F1$ Score, AP
	reward	feedback_time	feedback_time + 1.0	2	**Balanced Acc**, $F1$ Score, AP

Frame-Level Tasks. For the movement interval, we use a 1 second window of behavioral signals (`wheel_speed`, `left_paw_speed`, `right_paw_speed`, and `whisker_motion_energy`) starting from movement onset (defined as the beginning time of the first sustained change in wheel position greater than 0.1 radians after the quiescent period). Decoding these movement-oriented signals in a window aligned to first movement onset ensures robust variation and task-relevant changes in the target signals. Note that right paw speed may provide an easier target than left paw due to the orientation of the camera from which the data is derived, consistent with results in Table 1.

For licking rate, which occurs after the feedback phase (water is delivered via solenoid upon reward, or a noise burst is played over a speaker upon non-reward), we define a 1 second window starting at feedback onset.

Sequence-Level Tasks. For stimulus contrast, we decode the contrast level of the visual stimulus presented to the mouse, which can take one of five values: 100%, 25%, 12.5%, 6.25%, and 0%. We use a 100 ms window starting at stimulus onset (defined as the appearance of the Gabor patch on the screen), consistent with the decoding window used in (IBL et al., 2025b). This window is chosen carefully: a longer window risks leakage from the movement or reward phase, which could be spuriously correlated with higher stimulus contrast, since the mouse is more likely to move and be rewarded when the stimulus is more visible. At the same time, 100 ms provides sufficient neural signal to decode a response without overlapping substantially with reward and decision-making activity. We also hypothesize that reaction time may itself be correlated with stimulus contrast; using a fixed window rather than one that varies with reaction time therefore reduces the risk of spurious correlations. Because we use a 1 second window overall, the stimulus contrast window also includes 900 ms prior to stimulus onset, a period during which the screen is blank and no task-relevant information is present. We do not consider this a problem: this pre-stimulus baseline may aid decoding by providing a reference level of neural activity, and we expect models to be capable of selecting the relevant portion of the input even when additional context is provided.

For choice, we use a 1 second window ending at movement onset, ensuring that no information from the movement phase leaks into the decoding window. In (IBL et al., 2025b), this window was only 100 ms; we extend it to 1 second for standardization and apply the same reasoning as above: models are expected to be robust and to attend selectively to the task-relevant portion of the window.

For reward, we use a 1 second window starting at feedback onset, matching the licking rate window.

For sequence-level tasks we report the class distribution across all splits except the evaluation test set in Table A5.

Table A5: Class distribution of sequence-level tasks per split.

Split	**Choice**		**Contrast**					**Reward**	
	Right ($c=0$)	Left ($c=1$)	0% ($c=0$)	6% ($c=1$)	12.5% ($c=2$)	25% ($c=3$)	100% ($c=4$)	False ($c=0$)	True ($c=1$)
Pretrain	0.497	0.503	0.116	0.219	0.219	0.221	0.225	0.168	0.832
Eval Train	0.499	0.501	0.114	0.217	0.219	0.225	0.224	0.158	0.842
Eval Val	0.462	0.538	0.111	0.233	0.228	0.205	0.223	0.147	0.853

Several observations are worth noting. First, reward is a binary classification task and its distribution is heavily imbalanced: the mouse receives a reward in over 82% of trials across all splits. Second, choice is also binary and its distribution is approximately balanced ($\sim$50% per class), which reflects the trial structure: trials alternate between blocks (known as block priors) in which the stimulus appears on one side 80% of the time and on the other 20% of the time. Since these blocks are balanced across the session, the overall marginal distribution is near-uniform.

For stimulus contrast, the distribution across the 5 classes is approximately uniform across the four non-zero contrast levels ($\sim$22% each), but the 0% contrast class is underrepresented at 11%. This is due to our trial exclusion criteria, which remove trials with reaction times that are too long or with no wheel movement; mice also tend to respond less when no visual stimulus is present. We note that the 0% contrast condition is closely entangled with the block prior structure: in this condition, the mouse has no visual guidance and is expected to rely on its internalized block prior to decide which way to turn the wheel. This introduces a level of complexity that is out of scope for this benchmark and has been studied in depth in (Findling et al., 2025).

A Note on Window Overlap. Some windows can overlap (for example, the stimulus contrast and choice windows), but we do not consider excluding them as our goal is to maximize the number of valid trials while ensuring that all tasks are included within each trial. In extreme cases, when the mouse's reaction time is shorter than 100 ms (which occurs in 18% of trials of train and val split from the evaluation sessions), the choice window precedes the stimulus contrast window entirely.

Various Difficulty of Behavioral Targets. Some behavioral variables, such as reward or coarse movement signals, are relatively accessible, while others like licking rate remain substantially more difficult. This range reveals both where simple models are already sufficient and where large-scale pretraining provides meaningful gains.

Evaluation Paradigm. While pretraining can leverage both frame-level and sequence-level behavioral signals in a flexible setting (single- or multi-task, with optional modalities such as anatomical labels), we impose stricter constraints during downstream evaluation. All evaluation is performed on a single session and a single task at a time, without cross-session calibration. This reflects our target use case: a single pretrained model that generalizes to new animals and sessions unseen during pretraining. For the baselines we implemented are all calibrated using the train and validation splits of the evaluation session via fine-tuning. Note that our splits support further research in modified evaluation regimes, such as few-shot or zero-shot adaptation, through using a restricted set of the training split. All reported numbers correspond to single-task inference on the held-out test split of each session.

What Is Enforced, Recommended, and Left Open. We distinguish between three levels of constraint in the evaluation protocol.

Enforced. The test set is always the last 40% of trials in the session, reflecting the temporal structure of the data and ensuring consistency across all reported results. The decoding windows and task definitions described above are fixed and must not be modified. In TS1, these windows are sampled directly from the trial-aligned task intervals using the `TrialSampler` from `torch_brain`.

Recommended. The train and validation splits cover the first 40% and middle 20% of trials, respectively. When calibration is performed, we recommend fine-tuning with task-specific losses: cross-entropy for classification tasks, mean squared error (MSE) for continuous behavioral signals, and negative log-likelihood (NLL) under a Poisson distribution for licking rate. Regarding unit selection, we conducted a preliminary analysis to test the effect of unit filtering on decoding. Our initial results showed that, generally, the less filtering the better. We think this makes sense, since (1) decoders can learn to ignore units with poor signal, and (2) several of the QC filtering criteria are related to spike sorting, but even if a unit is not sorted correctly a decodeable signal may still be present. While we did not conduct a more extensive analysis with other baselines, we've generally observed this trend to be consistent. Hence, for all of the baselines we present in Table 1, we employ no unit filtering. That said, QC filtering remains a valid alternative, and benchmark users are free to experiment with filtering options that work best for their models.

Left open. The context window size and QC filtering are left open for the community to explore, during pretraining and evaluation. During evaluation, while the target window is fixed, the context window can be expanded from the default 1s up to 20s. The train and validation splits may also be reduced or removed entirely to probe few-shot or zero-shot generalization, provided the test split remains unchanged.

Split Statistics. Table A6 reports the size of each split under three tokenization schemes: spike tokens, used by event-based models such as POYO; unit-bin tokens, used by patch-based models such as NDT2; and time-step tokens, used by models that keep units in the channel dimension, such as NDT-Stitch. Token counts assume the default 1 s context window and 20 ms bins; expanding the context window as permitted above scales them proportionally. Spike tokens are given as a range because each task aligns its 1 s window to a different event, so the same number of windows contains different amounts of neural activity.

Baselines. For our single-session baselines, we fix the input to neural activity only (no anatomical labels or auxiliary signals), using the unit selection from each model's original configuration. We use the recommended train and validation splits and evaluate in the single-task inductive setting unless otherwise specified. We apply no unit filtering in TS1. All baselines in Table 1 are run on 5 seeds.

Metrics. We report task-specific metrics for each decoding target. For frame-level regression tasks (`whisker_motion_energy`, `wheel_speed`, `left_paw_speed`, and `right_paw_speed`), we report the coefficient of determination (R^2) and Pearson's r. For `licking_rate`, which follows a count distribution, we report the pseudo-R^2 measure D^2. For

Table A6: TS1 split statistics on the `all_units` build. Samples are 1 s task-aligned windows, disjoint within a task. Spike tokens count one token per spike; unit-bin tokens count one per (unit, 20 ms bin); time-step tokens count one per 20 ms bin with units in the channel dimension.

				Tokens		
Split	Units	Samples	Duration (h)	Spike	Unit-bin	Time-step
train	43,187	7,795	2.17	66.3 – 81.8 M	583.9 M	389,750
val	43,187	3,471	0.96	31.2 – 37.9 M	254.6 M	173,550
test	43,187	5,894	1.64	55.7 – 67.1 M	424.8 M	294,700

all sequence-level classification tasks (`stimulus_contrast`, `choice`, and `reward`), we report balanced accuracy and $F1$ score, accounting for the class imbalances noted in Table A5.

D.2 TS2 Details: Neural Activity Prediction

Table A7: Summary of pretraining evaluation task associated with TS2 (Neural Activity Prediction).

Task	Window Type	Interval Start	Interval End	Mask Ratio	Mask Dimension	Criterion	Metric
`co-smoothing`	rolling (20 ms)	`stim_on_time` − 1.0	`feedback_time` + 1.0	10%	neuron	random held-out†	D^2, bps
`forecasting`	rolling (20 ms)	`stim_on_time` − 1.0	`feedback_time` + 1.0	10%	time	deterministic last timestep	D^2, bps

†The held-out neuron mask used for evaluation is fixed and accessible only on the test set, to prevent optimization against the evaluation criterion during training.
Note: D^2 and bps are defined in Appendix E.

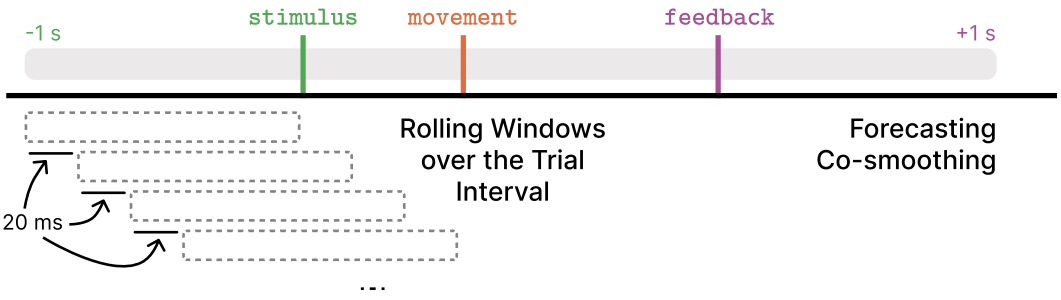


Figure A10: **Evaluation Windows for TS2.** Each trial spans from 1 s before stimulus onset to 1 s after feedback, with key events (stimulus, movement, feedback) marked along the timeline. Context windows of 1 s are extracted using a rolling procedure with a 20 ms stride across the full trial interval, and serve as inputs for both the co-smoothing and forecasting tasks.

Tasks. Task Suite 2 (TS2) evaluates models on neural activity prediction: the ability to reconstruct masked portions of population spiking activity from the observed remainder, without any behavioral supervision. TS2 comprises two complementary tasks (see Table A10). In *co-smoothing*, a fixed 10% of units per session are designated as held-out at data preparation time (using a fixed random seed). During inference, the model observes the remaining 90% of units and must predict the held-out units' spike counts. In *forecasting*, the model observes the first 90% of each context window and must predict the final 10% of time bins. Both tasks use 20 ms bins and a 1 s context window, yielding 50-bin inputs and 5-bin targets for forecasting, and a per-session unit mask for co-smoothing.

Splits and Provided Labels. TS2 uses a different temporal splitting strategy than TS1: trials are assigned to interleaved 5-minute blocks (see Figure A6 and Appendix C.2 for details). Unlike TS1, no behavioral labels are provided: the prediction targets are the binned spike counts of the held-out units (co-smoothing) or future time bins (forecasting), derived directly from the neural data.

Evaluation Paradigm. TS2 evaluation measures how well a model predicts unobserved neural activity within a context window sampled from the test split of each session. Evaluation windows

are extracted using a rolling procedure across the full trial interval, rather than being anchored to a specific task epoch, to avoid conflating prediction performance with the stereotyped and highly predictable neural dynamics that tend to dominate task-aligned windows. For the baselines reported here, models are fully fine-tuned on the training and validation splits of each session. All reported numbers are computed on windows drawn sequentially from the test split with a stride equal to the bin size 20 ms (see Figure A10).

What Is Enforced, Recommended, and Left Open. We distinguish between three levels of constraint in the evaluation protocol.

Enforced. The test split always corresponds to the designated test block, ensuring temporal consistency across methods. For co-smoothing, the held-out unit mask is fixed at data preparation time (10% of units per session, drawn with a fixed seed) and must not be modified. For forecasting, the masked portion is always the final 10% of the context window. The bin size (20 ms), the held-out masks, and the context window length at test time (1 s) must all remain unchanged. To ensure that only high-quality spikes contribute to the reconstruction target, both tasks are required to be evaluated with quality-controlled (QC) units (see Appendix A.

Recommended. We recommend training with a Poisson negative log-likelihood loss (with log-rate inputs).

Left open. The context window size during training is left open; our baselines use 1 s, but longer windows may benefit temporal forecasting.

Split Statistics. Table A8 reports the size of each split under three tokenization schemes: spike tokens, used by event-based models such as POYO; unit-bin tokens, used by patch-based models such as NDT2; and time-step tokens, used by models that keep units in the channel dimension, such as NDT-Stitch. Every baseline we report in TS2 uses the time-step scheme. Token counts assume the default 1 s context window and 20 ms bins. Unlike TS1, whose windows are disjoint trials, the validation and test samplers step through each session at 0.5 s and at one bin respectively, so their windows overlap and each spike is tokenized roughly twice at validation and thirty-five times at test. Task-specific masking reduces the input further at evaluation time.

Table A8: TS2 split statistics on the `selected_units` build. A sample is a 1 s window drawn by the split's sampler, and duration is the total span of the sampling intervals the sampler draws from. Spike tokens count one token per spike; unit-bin tokens count one per (unit, 20 ms bin); time-step tokens count one per 20 ms bin with units in the channel dimension. All three are counted as sampled, so overlapping windows count the same spike more than once.

				Tokens		
Split	Units	Samples	Duration (h)	Spike	Unit-bin	Time-step
train	3,778	19,033	6.05	28.7 M	122.5 M	951,650
val	3,778	16,952	2.57	25.8 M	109.8 M	847,600
test	3,778	700,471	5.59	1067.5 M	4529.5 M	35,023,550

Baselines. All TS2 baselines utilize neural activity as the sole input, without auxiliary signals or anatomical labels. An exception is MtM, which leverages brain region designations to define its masking schema. For pretrain model, we report results following full fine-tuning on the training and validation splits, with performance averaged across 5 random seeds.

Metrics. Both tasks are evaluated using two complementary metrics that assume a Poisson spiking model: the Fraction of Deviance Explained (D^2) and Bits Per Spike (bps). Both metrics are defined in detail in Appendix E.

D.3 TS3 Details: Neuron Identity Prediction

Tasks. Task Suite 3 (TS3) assesses whether models can infer anatomical position of neurons in a zero-shot manner. Such a task simultaneously probes representations for captured anatomical structure, and serves as a proxy for in vivo histology as a downstream application (Liu et al., 2021).

Table A9: Hyperparameters used for TS3 Linear probe. Implemented as `LogisticRegression` classifier from `sklearn`. No hyperparameter tuning is performed as the solver is deterministic given fixed inputs.

HYPERPARAMETER	VALUE
PREPROCESSING	STANDARDSCALER
SOLVER	LBFGS
INVERSE REGULARIZATION C	1.0
MAX ITERATIONS	1000
TOLERANCE	10^{-5}
CLASS WEIGHT	BALANCED

Splits and Provided Labels. TS3 is unique in that splits are defined over units instead of time. In particular, we aggregate all units over animals in the pretraining corpus to define the training set of brain region labels. Then the evaluation set forms the test labels.

Evaluation Paradigm. For a model to be evaluated on TS3, it must produce neuronal unit-level embeddings that can be probed on the brain region classification task. These embeddings are derived directly from neural activity, including spike trains used as input for TS1 and TS2, as well as metadata such as spike waveforms. Different architectures can represent neuronal identity in different ways, hence every model has its own designation for what constitutes unit embeddings. However, in the end, the model must be able to produce two sets of embeddings $E_{\text{train}} \in \mathbb{R}^{N_{\text{pretrain}} \times D}$ and $E_{\text{test}} \in \mathbb{R}^{N_{\text{eval}} \times D}$, where N_{pretrain} and N_{eval} are the number of units in the pretraining and evaluation corpora, respectively, and D is an embedding dimension defined by the model.

What is enforced in TS3? To ensure that only high-quality spikes contribute to the reconstruction target and that the neural identities are well-defined, TS3 requires high quality-controlled (QC) over units (Appendix A). Aside from this requirement, the mechanism by which embeddings are generated is fully flexible.

Types of Probes. Given embeddings, TS3 provides a standardized set of probes from which a brain region classifier can be fit. Following (Yu et al., 2025), these probes are organized along two axes:

1. Linear vs MLP: we either fit a Linear probe via the `LogisticRegression` classifier provided by `sklearn`, or train an MLP probe in `torch`. For the latter, we optimize hyperparameters using the Tree-structured Parzen Estimator (TPE) algorithm (Bergstra et al., 2011) implemented in Optuna (Akiba et al., 2019). Refer to Table A9 for details on the hyperparameters of the standard Linear probe, and Table A10 for the hyperparameter search space used for tuning the MLP probe.
2. Single- vs multi-unit: in (Yu et al., 2025) it was found that by aggregating consensus from nearby neurons, region classification accuracy can be improved. We implement this approach in a standardized evaluation pipeline for which we call the *multi-unit* probe. In particular, class predictions are averaged over all neurons in the vicinity of the target neuron, using probe depth to measure proximity.

Evaluation Regimes. Although the brain region classification task is designed to test zero-shot transfer with respect to region labels, models may vary in how they are able to generate embeddings. If a model requires gradients to generate embeddings on a new set of neurons, then it needs to be fine-tuned on the evaluation sessions using its pretraining task. Since then the model's weights were updated to calibrate on the new session, it has become exposed to the new neural population. We call this the *transductive zero-shot setting*. If, on the other hand, a model is able to generate embeddings in a forward pass without having to calibrate, we call this the *inductive zero-shot setting*.

Metrics. Due to imbalance in the region distribution (see Figure A3), we report macro-F1 score as the primary metric.

Table A10: Hyperparameter search space for TS3 MLP probe. Tuned for 100 trials with 4 concurrent trials at a time.

HYPERPARAMETER	SEARCH SPACE	SAMPLING TYPE
MODEL PARAMETERS		
BIN SIZE (MS)	{5, 10, 20, 40}	CATEGORICAL
DEPTH	{1, 2, 3}	DISCRETE
HIDDEN DIMENSION	{32, 64, 128, 256, 512}	DISCRETE ($\log_2$)
DROPOUT	{0.0, 0.2, 0.4, 0.6}	DISCRETE (STEP 0.2)
BATCH NORM	{TRUE, FALSE}	CATEGORICAL
TRAINING PARAMETERS		
NUMBER OF EPOCHS	{1000, 3000, 5000}	DISCRETE (STEP 2000)
BATCH SIZE	{64, 128, 256, 512, 1024}	DISCRETE ($\log_2$)
WEIGHT DECAY	$[10^{-6}, 10^{-1}]$	LOGUNIFORM
BASE LEARNING RATE†	$[10^{-6}, 10^{-2}]$	LOGUNIFORM

E METRICS AND RANKING

E.1 FRACTION OF DEVIANCE EXPLAINED UNDER POISSON MODEL (D^2)

Poisson negative log-likelihood. For a single observation with predicted log-rate ρ_i and observed spike count $n_i \in \mathbb{N}_{\geq 0}$, the Poisson negative log-likelihood (NLL) is

$$\ell(\rho_i;\, n_i) = e^{\rho_i} - n_i\, \rho_i + \ln(n_i!). \tag{1}$$

Summed over all T time bins, the total NLL of a model is $\mathcal{L} = \sum_{i=1}^{T} \ell(\rho_i;\, n_i)$.

Three reference models. Fraction of Deviance Explained (Cameron & Windmeijer, 1997; Xia et al., 2026) is defined with respect to three nested models.

1. **Predicted model.** The model of interest outputs log-rates $\{\rho_i\}$, giving
$$\mathcal{L}_{\text{pred}} = \sum_{i=1}^{T} \big(e^{\rho_i} - n_i\, \rho_i + \ln(n_i!)\big). \tag{2}$$
2. **Saturated model.** The ideal model sets $\hat{\lambda}_i = n_i$ exactly, achieving the minimum achievable NLL:
$$\mathcal{L}_{\text{sat}} = \sum_{i=1}^{T} \big(n_i - n_i \ln n_i + \ln(n_i!)\big), \tag{3}$$
3. **Null model.** The baseline predicts a constant rate equal to the empirical mean $\bar{n} = N/T$, where $N = \sum_{i=1}^{T} n_i$:
$$\mathcal{L}_{\text{null}} = \sum_{i=1}^{T} \big(\bar{n} - n_i \ln \bar{n} + \ln(n_i!)\big). \tag{4}$$

Fraction of Deviance Explained-D^2. The metric is defined as

$$D^2 = 1 - \frac{\mathcal{L}_{\text{pred}} - \mathcal{L}_{\text{sat}}}{\mathcal{L}_{\text{null}} - \mathcal{L}_{\text{sat}}}. \tag{5}$$

It equals 1 when the predicted model matches the saturated model, 0 when it matches the null model, and is negative when the model is worse than the null. Reported D^2 values are clipped to 0.

E.2 BITS PER SPIKE (BPS)

Bits per spike (bps) measures the log-likelihood gain of the predicted model over the null model, normalized by the total spike count and converted to base-2 bits:

$$\text{bps} = \frac{\mathcal{L}_{\text{null}} - \mathcal{L}_{\text{pred}}}{N_{\text{sp}} \ln 2}. \tag{6}$$

A positive bps indicates the model predicts spike timing more accurately than the mean-rate baseline; $\text{bps} = 0$ corresponds to the null model.

E.3 COEFFICIENT OF DETERMINATION (R^2)

For a continuous target with observations y_i, predictions $\hat{y}_i$, and mean $\bar{y} = \frac{1}{T}\sum_{i=1}^{T} y_i$, the coefficient of determination is

$$R^2 = 1 - \frac{\sum_{i=1}^{T}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{T}(y_i - \bar{y})^2}. \tag{7}$$

It equals 1 for a perfect prediction, 0 when the model matches the mean baseline, and is negative when the model is worse than the mean baseline. Reported R^2 values are clipped to 0.

Note that R^2 is the D^2 metric under a Gaussian observation model (McCullagh et al., 1989). Under this model, the notion of bits-per-spike (E.2) becomes proportional to MSE, and the saturated model is 0 (achieved under Gaussian noise). Hence, indeed D^2 reduces to R^2.

E.4 Rank Aggregation Procedure

To summarize performance across the many task-specific metrics reported within each task suite, we compute an average rank for each model that reflects statistical significance in point estimates. This follows the general practice of rank-based aggregation across evaluation sets (Demšar, 2006), gating rank assignment on pairwise significance testing rather than raw score differences alone (Dror et al., 2018). We treat each evaluation session as a separate dataset: for a fixed task and session, we rank models using the step-down procedure in Algorithm 1, and average the resulting ranks across sessions to obtain the per-task rankings reported in Tables A27, A31. These per-task rankings are in turn averaged across tasks to obtain the overall average rank reported in Tables 1 and 2.

For each model, we take the point estimate to be the mean score over 5 random seeds, and use the standard error of the mean (SEM) over seeds to quantify variance for the significance test in Algorithm 1.

Algorithm 1 Step-down rank assignment for a single task and session

Require: Models $\{m_1, \ldots, m_K\}$ with per-seed scores; significance level α
1: Compute mean score $\bar{s}_k$ and SEM σ_k over seeds for each model m_k
2: Sort models by $\bar{s}_k$ so that $m_{(1)}$ is best-performing; let π denote this order
3: rank$[\pi(1)] \leftarrow 1$
4: $a \leftarrow \pi(1)$ ▷ current anchor
5: **for** $i = 2$ **to** K **do**
6: $m \leftarrow \pi(i)$
7: $p \leftarrow$ one-sided Welch's t-test, H_1: anchor a outperforms m
8: **if** $p \geq \alpha$ **then**
9: rank$[m] \leftarrow$ rank$[a]$ ▷ not significantly worse than the anchor
10: **else**
11: rank$[m] \leftarrow i$ ▷ standard competition ("1224") ranking
12: $a \leftarrow m$ ▷ m becomes the new anchor
13: **end if**
14: **end for**
15: **return** rank$[m_1], \ldots,$ rank$[m_K]$

Each comparison is made only against the current anchor rather than against all previously ranked models; a model is assigned a new, strictly lower rank only once it is shown to be significantly worse than the best remaining anchor. This anchor-based, sequential structure means a group of models found statistically indistinguishable from one another all receive the same rank, and the next distinct rank is set to the group's position in the sorted order (line 10) rather than incrementing by one (e.g., a tied pair at rank 2 is followed by a rank of 4, not 3). This grouping behavior parallels compact letter displays used to summarize sets of pairwise comparisons in applied statistics (Piepho, 2004), though we note the same caveat that applies to that literature: a shared rank here reflects a failure to reject the null hypothesis of no difference at α, not demonstrated equivalence between models.

We assign the tied rank itself using standard competition ("1224") ranking, computed via `scipy.stats.rankdata(..., method='min')` (Virtanen et al., 2020), rather than the fractional (mid-rank) ranking conventionally used when averaging ranks in the style of (Demšar, 2006). We make this choice because competition ranking directly encodes the quantity of interest for this benchmark: a model's rank equals one plus the number of models shown to be significantly better than it. Fractional ranking would instead average a tied group into the ranks its members would have occupied absent ties, which can obscure the significance information. We use $\alpha = 0.05$ throughout. Algorithm 1 is applied independently per task and session; the resulting ranks are averaged first over sessions (Appendix I) and then over tasks to yield the average rank columns reported in the main text.

F Overview of Baseline Models

We implement a panel of baselines reflecting two complementary goals. First, we establish task-specific reference points using specialized single-session baselines, which serve as points of comparison on individual tasks. Second, we pretrain typical neural data methods on a shared corpus and evaluate them downstream on applicable tasks, providing a starting point for the community to compare approaches and identify gaps.

In the following sections, we provide background on the various aspects of design involved in constructing models for neural data. These include tokenization schemes or input processing, model architectures, and training objectives. In Section G, we describe how we defined and tuned hyperparameters for the different task suites, including single-session baselines, pretraining, and finetuning on individual tasks.

F.1 Tokenization Schemes

Models of neural data can accept various modalities of input, including spike trains (encoding exact timing information along with neuron identity), spike waveforms, and local field potential (LFP). A major part of the design space involves choosing how input data should be structured while the model extracts representations, in such a way that is compatible with their choice of architecture, training objective, and underlying assumptions. This choice is often denoted *tokenization*, popularized by its use in other Machine Learning domains such as Natural Language Processing (Sennrich et al., 2016) and Computer Vision (Dosovitskiy et al., 2020) to indicate the fundamental units of information as seen by the model.

F.1.1 Binning and Patching

A natural choice for representing neural spike trains is to convert the discrete event sequences into bin counts in a regularly spaced grid. The idea is to derive an approximation of the instantaneous firing rate, motivated by the assumption that neural information is carried via rate coding (Adrian, 1926). The choice of bin size modulates the tradeoff between smooth rate estimates and precise timing.

Given a raster of T bin counts over a fixed context length from N neurons, the next choice is how to ingest this spatiotemporal grid into a model. One idea is to flatten the bins into $N \cdot T$ tokens, treating them all the same throughout computations. This approach loses precise temporal and spatial information unless it is explicitly included via positional information, and could result in large computational complexity given many neurons. At the same time, this approach introduces minimal inductive biases and is a typical choice for simple baselines such as Linear and MLP.

Several works have explored alternative tokenization schemes given binned spikes. NDT (Ye & Pandarinath, 2021) extracts tokens as temporal *patches* by first slicing multiple timestamps at a time, then projecting the spatial dimension into a shared embedding space. STNDT (Le & Shlizerman, 2022) separately tokenizes spatial and temporal slices, applies independent attention modules, and then fuses the features. NDT2 (Ye et al., 2023) adopts a ViT (Dosovitskiy et al., 2020) style tokenization scheme by treating the $N \times T$ raster as an image and applying both spatial and temporal patching. Different approaches yield tradeoffs in terms of computational complexity and spatial or temporal resolution throughout the architecture.

F.1.2 Spike Tokenization

An alternative philosophy is to retain exact timing information throughout computations, motivated by the idea that neural information could be carried via temporal coding (Thorpe & Gautrais, 1998). From the modeling perspective, this approach treats every unit of communication between neurons, i.e. exact spike times along with the identity of the neuron, as a token. Such an approach was introduced by POYO (Azabou et al., 2024) and extended to follow up works (Ryoo et al., 2025). The advantage of this tokenization scheme is that it is not reliant on a fixed bin size, considering optimal temporal resolutions can vary between neural circuits and some circuits may even exhibit temporal multiplexing at multiple timescales (Panzeri et al., 2010). On the other hand, such an approach requires unit-level tokenization resulting in very long token sequences, especially with a large number of neurons or very active neurons.

F.1.3 INTER-SPIKE INTERVAL (ISI) DISTRIBUTIONS

Another perspective is that information lies in relative timing, which can be captured by the inter-spike interval (ISI) distribution (Perkel et al., 1967). For a given neuron's spike train, its ISIs are computed as the time differences between consecutive spikes within a recording window and then binned into a histogram. The choice of binning approach should aim to distribute the probability mass of the ISIs uniformly across bins. ISI distributions are typically heavily right-skewed.

A linearly spaced binning approach, which assigns equal resolution across the entire ISI range, concentrates the majority of the probability mass into the first few bins, creating a highly non-uniform distribution that obscures fine-grained dynamics. Conversely, log-spaced binning grants finer resolution to shorter intervals, resulting in a more uniform representation that preserves the fine-grained temporal precision of short-interval events (Dorval, 2008).

The recording window length introduces another tradeoff. A longer window, such as a per-session window, can provide a more stable estimate of the ISI distribution as it contains more spikes, and therefore more intervals (Gerstner et al., 2014). However, because a per-session window pools ISIs across an entire session, it may obscure fine-grained local temporal dynamics. In contrast, per-trial or fixed-context windows can better preserve local spiking patterns, though they may yield noisier ISI estimates.

F.1.4 AUTOCORRELOGRAMS (ACG)

Similar to the ISI distribution, an autocorrelogram (ACG) also captures relative timing information by treating each spike as a reference event and constructing a histogram of pairwise time lags to other spikes from the same neuron (Perkel et al., 1967). Unlike the ISI distribution, which only considers intervals between consecutive spikes, the ACG includes delays between every pair of spikes within a fixed lag window. This allows the ACG to capture repeated temporal patterns in the neuron's spike train.

As with ISI histograms, the lag bins may be linearly or logarithmically spaced, with log-spaced bins preserving finer resolution at short lags while compressing longer lags into broader bins (Dorval, 2008). This is useful because bins near zero lag can capture closely spaced spike pairs, including burst firing patterns and refractory-period effects.

F.1.5 SPIKE WAVEFORMS

Neural activity can also be represented through spike waveforms, where each waveform captures the extracellular action potential (EAP) measured by extracellular recording electrodes during a short time window around each detected spike (Gold et al., 2006). Whereas tokenization schemes such as ISI distributions and ACGs capture spike timing, EAP waveforms encode the shape of the recorded spike itself.

The shape of a spike waveform is often characterized by a prominent negative and positive peak, reflecting the extracellular signature of a neuron's action potential (Gold et al., 2006). Because this waveform shape can vary greatly across neuron classes, common neuroscience problems such as cell-type classification often use waveform-derived features, such as waveform width and peak-to-trough amplitude (Yu et al., 2025).

F.2 MODELS

F.2.1 LINEAR AND MULTI-LAYER PERCEPTRON (MLP)

A straightforward baseline for decoding involves mapping flattened bin counts directly into the target readout. Due to the complexity of loss functions for certain decoding tasks (e.g., masked MSE for paw kinematics, Poisson NLL for licking rate), we standardized training linear models using gradient descent.

We can introduce nonlinearity into the mapping from flattened bins to target readout via a Multi-Layer Perceptron (MLP). Note that in initial experiments with the MLP, we found this fully flattened binned counts representation performed better than taking temporal or spatial slices. These baselines

introduce very little inductive bias on the input and output structures, which can make them flexible but difficult to train with limited data.

See Appendix G.1.1 and G.1.2 for hyperparameter selection of Linear and MLP models, respectively.

F.2.2 Convolutional Neural Network (CNN)

A standard approach for extracting local, translation-invariant temporal features is the Convolutional Neural Network (CNN). We start with a typical implementation of convolutional layers on time-series data using the Temporal Convolutional Network (TCN), which achieves wide receptive fields via stacked dilated convolutions (Lea et al., 2016). However, we retain a dilation factor of 1 for simplicity while defining the search space. Input features are temporal slices over a binned raster, where the bin size is a model hyperparameter. The result is a temporal resolution that is directly modulated by the bin size throughout the model architecture. To support both sequence- and frame-level tasks, we attach adaptive average pooling before a final linear readout.

In the CNN, convolutions are interleaved with nonlinear activations providing a structure similar to the MLP but with strong inductive biases of locality and translation invariance along the temporal axis. Note, however, that if the activations were replaced with *identities*, the convolutions would collapse to a single linear map from a fixed context window of binned counts to the readout.

A key benefit of this representation is that the CNN then expresses a standard linear baseline for neural decoding of continuous variables, wherein a short context window is used to readout the next timestep in an autoregressive fashion. For example, in the IBL BrainwideMap (IBL et al., 2025b), continuous targets like whisker motion energy and wheel speed are regressed in this way using a 200ms sliding context window of population activity. The CNN thus strictly contains this linear baseline as a special case.

See Appendix G.1.3 for more details on hyperparameter selection.

F.2.3 Gated Recurrent Unit (GRU)

Recurrent architectures offer a natural inductive bias for sequential neural data by maintaining a hidden state that is updated at each timestep, implicitly integrating information over arbitrary temporal horizons. The earliest such model, the Recurrent Neural Network (RNN) (Elman, 1990), suffers from well-known training instabilities: gradients either vanish or explode when backpropagated through long sequences, making it difficult to capture dependencies beyond a short effective context window (Bengio et al., 1994). The Long Short-Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997) addressed this by introducing a dedicated cell state governed by input, forget, and output gates, which create additive gradient paths that resist vanishing. The Gated Recurrent Unit (GRU) (Cho et al., 2014) simplified this design by merging the cell and hidden states into a single vector controlled by just two gates, a reset gate and an update gate, retaining comparable expressive power at reduced computational cost.

This gating mechanism is closely related to State Space Models (SSMs) (Gu et al., 2022), as both learn the transition matrices of a linear dynamical system, with gating providing a data-dependent, nonlinear modulation of the dynamics. The SSM perspective has a long tradition in systems neuroscience, where latent linear dynamical systems and their nonlinear extensions have been used to infer low-dimensional population dynamics from high-dimensional spike trains (Yu et al., 2008; Macke et al., 2011), including approaches that couple Gaussian Process priors with RNN transitions for smooth latent inference (She & Wu, 2020). The broader sequence-modelling literature has revisited linear recurrences for efficient long-range modelling (e.g., S4 (Gu et al., 2022), Mamba (Gu & Dao, 2024)), a trend that has begun to influence neural decoding through architectures that exploit their fast inference properties (see Section F.2.10).

As with the CNN, input features are temporal slices over a binned raster with bin size as a hyperparameter. Binned spike counts are first projected to a fixed hidden dimension via a linear layer, then processed sequentially by a multi-layer GRU. To support both sequence- and frame-level tasks, adaptive average pooling is applied over the temporal dimension before a final linear readout.

See Appendix G.1.4 for hyperparameter selection of the GRU model.

F.2.4 TS2 STATISTICAL BASELINES

Alongside the trained single-session baselines, TS2 includes a panel of statistical baselines: closed-form, zero-parameter methods that fit a handful of summary statistics on the training windows and select one or two hyperparameters on validation. Nothing is optimized by gradient descent, so these serve as a floor that quantifies how much of each TS2 task is solved by stationary firing statistics, instantaneous population coupling, or simple linear extrapolation, before any learned representation is involved. All methods emit log-rates and are scored with the same Poisson metrics as the trained models.

The two TS2 tasks hold out along orthogonal axes: co-smoothing holds out a subset of units and leaves the rest of the population simultaneously observed, whereas forecasting holds out the trailing timesteps and leaves only each unit's own past. A given method exploits exactly one of these structures, so each is declared for one task. Every method reads only observed entries, making them leakage-free by construction.

Co-smoothing methods.

- **Population coupling.** A rank-one model in which each unit's rate is modulated by a single global population signal, $r_i(t) = \bar{r}_i\, p(t)^\gamma$, where $p(t)$ is a Gaussian-smoothed, mean-one normalized trace of the observed population activity and γ controls the strength of coupling. The smoothing width σ and the exponent γ are selected on validation, and $\gamma = 0$ recovers the mean rate. This tests whether a single shared gain, with no unit-specific structure, explains the held-out activity.
- **RRR.** A reduced-rank ridge regression mapping the observed units to the held-out units at each timestep, with the rank k and the ridge strength λ selected on validation. Unlike population coupling, this captures unit-specific, multi-dimensional structure in the instantaneous population code, and it is the natural linear reference point for what a learned latent variable model must beat.
- **RRR (w/ ISI features).** The same rank-constrained ridge regression, but with each observed unit contributing, in addition to its binned count, two timing features per bin: the time since its last spike and its most recent inter-spike interval (see Appendix F.1.3), computed from the raw spike times. Because the regressors now mix counts with seconds, features are standardized before the solve. This isolates the value of the sub-bin timing information that binning discards, holding the regression itself fixed.

For both reduced-rank readouts, the held-out set used at test is disjoint from the observed set, so predicting one from the other is leakage-free. Hyperparameters are selected on an independent validation hold-out draw and the readout is then refit for the test draw, so the scored units take no part in selection.

Forecasting methods.

- **Trailing mean.** Each unit's horizon is predicted by the average of its own last K observed bins, held flat across the forecast window, with K selected on validation. This measures whether recent activity alone, without reference to session-level statistics, carries predictive signal about the immediate future.
- **Shrinkage.** A blend of each unit's trial-local rate with its global mean, $r_i = (1-\alpha)\,\bar{r}_i + \alpha\,\ell_i$, where ℓ_i is the mean over the observed portion of the current window and α is selected on validation. The two endpoints are exactly the mean rate ($\alpha = 0$) and the trailing mean over the full observed window ($\alpha = 1$), so an interior optimum indicates that trial-to-trial rate fluctuations are informative but noisy enough to require regularization toward the session average.
- **Ridge autoregression.** A per-unit ridge regression from the last L observed bins to each forecast bin, solved separately for every bin in the horizon, with L and λ selected on validation. Because each future bin gets its own weights, this captures the recent trend of a unit's rate rather than only its level, making it the strongest purely linear, single-unit forecaster in the panel.

See Appendix G.2.1 for the search grids and selection protocol used for each method.

F.2.5 AUTOENCODER (AE)

The Autoencoder (AE) extends the MLP baseline toward latent variable modeling: rather than mapping directly from input to readout, it introduces a bottleneck that encourages the network to capture low-dimensional structure in the population response. The model is trained to reconstruct its own input under a Poisson negative log-likelihood loss, treating the decoder outputs as log-rates of a Poisson spike-count distribution. Following the same tokenization scheme as the Linear and MLP baselines (Section F.2.1), the input is flattened and passed through an encoder-decoder MLP pair. The architecture is a symmetric counterpart to the MLP: the encoder projects the representation down by a factor of 2 at each layer, the decoder mirrors this by projecting back up by a factor of 2, and a final linear readout maps the reconstruction to the full output dimensionality. This symmetric bottleneck can be seen as a nonlinear generalization of PCA applied jointly across time and neurons.

F.2.6 LFADS

Latent Factor Analysis via Dynamical Systems (LFADS) (Pandarinath et al., 2018b) is a sequential variational autoencoder that models a trial of population activity as the output of a low-dimensional nonlinear dynamical system. A bidirectional GRU encoder maps the binned spike raster to a posterior over the generator's initial condition, and a second bidirectional encoder feeds a controller GRU that infers a time-varying input sequence, capturing structure the autonomous dynamics cannot produce. A generator GRU is rolled out from the sampled initial condition under these inferred inputs, projected to a small set of latent factors, and read out linearly to per-unit Poisson log-rates. Training maximizes an ELBO with a Gaussian prior on the initial condition and an autoregressive prior on the inferred inputs, the latter keeping those inputs temporally smooth rather than pushing them toward white noise.

We follow AutoLFADS (Keshtkaran et al., 2022) in treating the regularizers as search parameters rather than fixed constants: the two KL weights, the L2 penalties on the generator and controller recurrent weights, dropout, and the coordinated dropout rate are all tuned per session and task. Coordinated dropout, which corrupts a random subset of input entries and grades the reconstruction only on them, is what blocks the identity solution available to an autoencoder with a per-unit readout. For co-smoothing we additionally blank whole unit rows at a tuned rate, matching the shape the hold-out takes at evaluation. For forecasting, the reconstruction gradient is taken either from the held-out tail alone or from the whole window, which is itself a tuned choice.

See Appendix G.2.4 for hyperparameter selection.

F.2.7 LOLCAT

Local Latent Concatenated Attention (LOLCAT) (Schneider et al., 2023a) is a supervised model predicting neuron identity from neuronal activity timing data. Originally, it was used to predict a neuron's transcriptomic cell-type, but we are using it for brain region classification. For each neuron, spike trains within trial windows are converted into inter-spike interval (ISI) (see Appendix F.1.3) histograms using log-spaced bins. These histograms are passed through a per-trial encoder (i.e., MLP), and then aggregated across trials into a single per-neuron embedding via multi-head attention pooling. Finally, an MLP classifier is used to predict the brain region of each neuron.

To address class imbalance, LOLCAT utilizes an adaptive sampler that reweights classes during training. The sampler maintains class-specific sampling factors that are updated at each validation epoch using class-wise training and validation losses. It computes two scores: an overfitting score, which quantifies overfitting for a given class, and an undertraining score, which quantifies how well the model is performing on that class relative to the entire dataset.

LOLCAT also implements adaptive trial dropout, with the probability of dropping a trial being proportional to a specified hyperparameter p and inversely proportional to the neuron's firing rate in that trial.

See Appendix G.3.2 for more details on the implementation and hyperparameters used

F.2.8 CEBRA

CEBRA (Schneider et al., 2023b) is a self-supervised method that learns low-dimensional embeddings of neural data by contrasting neural activity against behavioral labels. We use the single-session variant of CEBRA, fitting a separate model for each session and training a MLP probe on the resulting embeddings to predict behavior. We do not use multi-session CEBRA, as it is incompatible with the `ts1` evaluation protocol: multi-session CEBRA requires all target sessions to be included during model fitting, and provides no standard mechanism for transferring pretrained model to held-out sessions. CEBRA supports three contrastive modes: a time mode, which contrasts neural activity based on temporal proximity alone; a behavior mode, which contrasts neural activity against behavioral labels; and a hybrid mode, which combines both time-contrastive and behavior-contrastive objectives. We treat the contrastive mode as a hyperparameter and select it per task and per session based on validation performance. In practice, we find that behavior mode is often detrimental for sequence-level tasks, whereas hybrid mode, which uses both temporal structure and behavioral supervision, tends to perform better on frame level tasks. Refer to Appendix G.1.5 for hyperparameter selection of the CEBRA model.

F.2.9 POYO

POYO (Azabou et al., 2024) is a method for pretraining on a large amount of neural spiking activity. It introduces spike tokenization (see Appendix F.1.2) in a Transformer-based backbone. In particular, it uses a PerceiverIO (Jaegle et al., 2021) backbone featuring a cross-attention layer on model inputs to compress the input token sequence (which can be quite long with a large number of units) into a relatively small latent token sequence. Following this compression, the operations no longer scale quadratically in time and space with the input size, allowing memory-efficient processing of individual spikes. The most expensive operation is the first cross-attention layer, hence computational budgets are primarily allocated with this initial layer in mind. The architecture uses a final cross-attention decoder to readout the target sequence. POYO is typically trained in a supervised fashion, where the target readout specifies the query sequence in the cross-attention decoder.

In the single-session, single-task case, we adapt the standard implementation (Azabou et al., 2024) into our evaluation pipeline. In particular, we tokenize input spikes as (u_i, t_i), where u_i represents the unit identity mapping to a lookup table containing *unit embedding*, and t_i represents timing information relative to the context window. This input token sequence is compressed into the latent sequence via an input cross-attention layer (where learnable latent tokens are queries and the input tokens are keys/values), followed by several layers of self-attention on the latent sequence. On the decoder, a learnable *session embedding* is repeated over the target sequence (e.g., the number of timestamps for frame-level tasks, or a single timestamp at the middle of the context window for sequence-level) to form the query sequence for a cross-attention decoder, where the latent tokens outputted from the final self-attention layer form the keys/values. After the final cross-attention layer the output tokens are projected via a final linear layer into the target dimension, depending on the task. POYO uses Rotary Position Embeddings (RoPE) (Su et al., 2024) to incorporate timing information into attention layers.

Note that in the single-session case, the set of learnable session embeddings is a singleton, hence the decoder is simply performing attention pooling where the session embedding is the learnable query. The notion of session embeddings becomes relevant in the multi-session setting. In the context of our benchmark, we only use multi-task POYO+ (Azabou et al., 2025) for unified pretraining that can be applied downstream to any of the decoding tasks, and single-session POYO for tuning and for finetuning. The multi-session, single-task POYO variant is not relevant to the benchmark since it does not fit within our evaluation protocol, and in pretraining it cannot produce a single model that can be applied to any task.

Still, to benchmark how the multi-task variant compares to single-task variants on each task, we pretrained POYO single-task (POYO-ST) and finetuned on each evaluation session in a cursory study. This study mirrors experiments done in the POYO+ (Azabou et al., 2025) paper on the Allen Institute's Brain Observatory dataset (de Vries et al., 2020), where the multi-task model is compared to single-task variants to observe the effect of multi-task learning. See Appendix A36 for results and discussion on this study on our benchmark tasks and splits.

Note that when finetuning POYO+ on the evaluation sessions, we do retain the task-specific head and learnable task embeddings used in POYO+ pretraining, but restricted to the specific task at hand. This is an architectural addition to what is used for the single-session POYO baseline necessary to effectively finetune the pretrained variant. See Appendix G.4.1 for more details on the finetuning implementation and hyperparameters used.

F.2.10 POSSM

POSSM (Ryoo et al., 2025) is a hybrid architecture for real-time, causal neural decoding that pairs POYO-style spike tokenization and input/output cross-attention with a recurrent state-space model (SSM) backbone. The key departure from POYO is that the self-attention layers over the full latent sequence are replaced with a recurrent backbone that operates over short, contiguous, non-overlapping time chunks (typically 50 ms). The input cross-attention operates within each chunk, compressing variable-length spike token sequences (u_i, t_i) into a single latent vector $z^{(t)}$, with RoPE (Su et al., 2024) encoding spike timing relative to the chunk and unit embeddings encoding unit identity, exactly as in POYO. This latent is then passed to the recurrent backbone whose hidden state $h^{(t)} = f_{\text{SSM}}(z^{(t)}, h^{(t-1)})$ propagates information across chunks. This enables constant-time updates as neural activity is streamed in, in contrast to POYO which reprocesses an entire context window at each new prediction. The output cross-attention decoder mirrors POYO's: queries built from a repeated learnable *session embedding* combined with target timestamps (via RoPE) attend over the k most recent hidden states ($k = 3$ in the original work) to produce behavioral predictions, preserving POYO's flexibility to emit multiple, irregularly-spaced targets per chunk and to query timestamps beyond the current input chunk.

We adapt the standard implementation (Ryoo et al., 2025) into our evaluation pipeline, with the only modification being that we use a non-causal, bi-directional recurrent backbone in place of the original uni-directional one to match other non-causal models used in TS1; all other architectural and tokenization choices follow (Ryoo et al., 2025) exactly. In the single-session, single-task case, the set of learnable session embeddings is again a singleton and the decoder reduces to attention pooling, analogously to single-session POYO. For cross-session/cross-dataset pretraining, the unit and session embedding tables span all training sessions, yielding the pretrained *o-POSSM* model that can be applied downstream to any of the benchmark decoding tasks. As with POYO+, the multi-session, multi-task pretrained variant produces a single model applicable to any task in the benchmark.

See Appendix G.4.2 for more details on POSSM finetuning implementation and hyperparameters used.

F.2.11 NDT

The Neural Data Transformer (NDT) (Ye & Pandarinath, 2021) is a standard transformer model operating on binned spikes. The original method was designed for latent variable modeling (Pei et al., 2021) where input spike trains are encoded into latent factors that summarize the neural population codes. These latent factors can then be used to produce predicted firing rates and decode behavior. Hence, the original method is inspired by BERT encoder (Devlin et al., 2019) and used masked modeling as a training objective. See Figure A14 for a comparison of masking schemes used by various models that employ masked modeling on bins.

Our treatment of NDT goes beyond the training objective to consider the architecture as a fundamental approach for processing binned spike tokens in a transformer. Drawing on the comparison of tokenization schemes in Appendix F.1, we consider a supervised variant of NDT (NDT Supervised) that can be directly used to perform on a downstream task of TS1, and as a single-session baseline it can be compared to the single-session POYO variant forming the two fundamental transformer baselines in TS1 (see Table 1).

NDT is also used as a single-session baseline in TS2, but there we use the original BERT-inspired masking scheme for latent variable modelling, which can then be evaluated on both co-smoothing and forecasting tasks. Masking is applied along the temporal dimension across all neurons simultaneously, such that contiguous blocks of timesteps are masked from the model. Masked bins are replaced by one of three augmentations sampled during training: the spike count is zeroed out, replaced by a random value, or left unchanged, following the token corruption strategy of BERT (Devlin et al., 2019).

Finally, we pretrain NDT (NDT-stitch) with its masked modeling objective across all pretraining sessions with downstream evaluation on TS1 and TS2. To generalize across sessions and animals that have varying numbers of units, common in practice due to spike sorting, we adopt the session-specific stitcher technique introduced in (Ye et al., 2023; Zhang et al., 2024). The stitcher is a lightweight linear or MLP layer that projects the variable neuron counts of each session into a fixed-dimensional latent space, allowing the transformer backbone to share weights across all sessions while the stitcher weights remain session-specific. When transferring to a new session or animal during fine-tuning, a new stitcher is initialized from scratch while the pretrained backbone is retained.

For further details on hyperparameter selection, see Appendix G.1.7 for the supervised single-session NDT baseline (TS1), Appendix G.2.3 for the single-session NDT baseline (TS2), and Appendix G.4.3 for the pretrained NDT-stitch model.

F.2.12 MtM

Multi-task masking (MtM) (Zhang et al., 2024) is a self-supervised transformer model for neural activity reconstruction. MtM uses the same architecture as NDT, but replaces the standard random timestep masking objective with multiple masking strategies designed to capture the spatiotemporal structure of neural population activity (see Figure A14). These objectives include: (1) causal masking, which masks future time steps and predicts them from past activity, corresponding to forecasting in TS2; (2) neuron masking, which randomly masks neurons and reconstructs them from the remaining neurons, corresponding to co-smoothing in TS2; (3) intra-region masking, which masks randomly sampled neurons within a randomly selected brain region and reconstructs them using unmasked neurons from the same region; and (4) inter-region masking, which masks all neurons within a brain region and reconstructs them using neurons from other regions. For each masking scheme, we apply input masking by zeroing out the masked portions of the data. Each of these masking schemes teaches the model about different structure in neural populations. During training, the model alternates across these objectives, enabling it to learn latent neural representations that support multiple inference-time tasks. In addition, we prepend a learnable prompt token to the transformer to indicate the masking scheme on which the model is being trained. This token can also be provided at test time to adapt the model to different downstream tasks.

We pretrain MtM with its designed masked modeling objective across all pretraining sessions with downstream evaluation on TS1 and TS2. We evaluate transfer both within the pretraining distribution (TS2) and in a more challenging out-of-distribution setting (TS1). For multi-session pretraining, we prepend a session identity token to indicate the session from which the neural data originates. To accommodate varying neuron counts across sessions, we adopt a similar strategy to NDT-Stitch. Specifically, we use a session-specific linear read-in layer to map neural activity from each session into a shared hidden dimension, followed by a shared MLP applied across all sessions (this shared MLP differs from our NDT-Stitch implementation). We additionally employ session-specific read-out stitchers that map the learned latent representations back to the original neuron count of each session.

See Appendix G.4.4 for more details on MtM finetuning implementation and hyperparameters used.

F.2.13 NuCLR

NuCLR (Arora et al., 2026) is a self-supervised framework for learning neuron-level representations from large-scale neural population activity, designed to capture identity-relevant attributes such as cell type and brain region. NuCLR produces a representation *per neuron* directly from population activity, without assuming any fixed neuron ordering or requiring session-specific alignment. Inputs are constructed by binning each neuron's spike train at a fixed bin-size (20 ms by default) and partitioning the bins into non-overlapping temporal patches of length T_{patch}, which are linearly projected into a D-dimensional latent space to yield a token sequence per neuron. These tokens are then processed by a spatiotemporal transformer that alternates between two types of attention: *temporal attention layers* that operate independently on each neuron's patch sequence (with RoPE (Su et al., 2024) encoding relative patch timing), and *spatial attention layers* that, at each time index, attend across neurons in the population to inject population-level context into each neuron's representation. The model uses L_T purely temporal layers followed by L_{ST} spatiotemporal layers (each combining a spatial block with a temporal block), and finally mean-pools each neuron's tokens over the temporal axis to yield a fixed-dimensional vector per neuron. The architecture is permutation-equivariant across neurons

and accepts populations of arbitrary size, allowing the same encoder to be applied to recordings with different numbers of neurons.

NuCLR is trained with a sample-wise contrastive objective inspired by SimCLR (Chen et al., 2020). Two temporally-spaced views of the same population are sampled within $\Delta T_{\max}$ of each other and independently subjected to neuron dropout (up to 50%) to encourage robustness to partial observations. Both views are passed through the spatiotemporal encoder and a projection head $g(\cdot)$, and an InfoNCE-style loss treats representations of the same neuron across the two views as positives while treating all other neurons in the same population as negatives. Crucially, negatives are restricted to within-recording neurons (and, for electrophysiology, within a single probe insertion) to avoid the trivial-negative problem that arises when neurons from unrelated recordings are pooled. We adapt the standard implementation (Arora et al., 2026) into our evaluation pipeline without modification. Following the linear-evaluation protocol used in the original work, we freeze the pretrained encoder, compute a single representation per neuron by averaging encoder outputs over windows sampled from each session, and train a linear classifier on top for downstream cell-type and brain-region classification.

F.2.14 NEMO

Neuronal Embeddings via Multimodal contrastive learning (NEMO) (Yu et al., 2025) is a multimodal contrastive learning framework for learning representations of individual neurons from neurophysiological data. Specifically, it jointly embeds two complementary views of the same neuron: activity autocorrelograms (ACGs) and average extracellular waveforms (see appendices F.1.4 and F.1.5). This method is motivated by the assumption that these two modalities carry shared information that is more discriminative of neuron identity than either modality alone, yielding representations useful for downstream cell-type and brain region classification tasks.

To get individual ACG and waveform representations, NEMO utilizes separate encoders for the two modalities: a 2-layer CNN for ACGs and a 2-layer MLP for waveforms. Then, both representations are projected into a shared embedding space via linear projection heads to a common dimensionality. A CLIP-style contrastive loss (Radford et al., 2021) is used to treat each unit's (waveform, ACG) pair as a positive pair and all cross-unit pairs as negatives. For downstream evaluation, the projection heads are discarded, and the pre-projection encoder representations are concatenated, with the ACG representation first, to produce a final unit embedding. The resulting embeddings are then evaluated in a supervised manner via linear and MLP probing for cell-type and brain region classification.

The original NEMO model applies modality-specific data augmentations: ACG images are augmented with temporal Gaussian smoothing, temporal jitter, amplitude scaling, additive Gaussian noise, and multiplicative pepper noise, whereas waveform templates use additive Gaussian noise.

See Appendix G.3.3 for more details on the implementation and hyperparameters used.

F.2.15 NDT2

NDT2 (Ye et al., 2023) is a transformer model for multi-context pretraining on binned neural spiking activity. It adopts an asymmetric encoder-decoder architecture: a large transformer encoder processes the masked spike bin sequence into latent representations, while a lightweight 2-layer transformer decoder reconstructs the masked tokens. This asymmetry concentrates representational capacity in the encoder while keeping the reconstruction decoder computationally cheap, following the design philosophy of masked autoencoders (He et al., 2022). Prior to encoding, spike bins are grouped into patches along the temporal dimension, compressing the representation spatially and reducing the effective sequence length seen by the encoder (analogously to the patch embedding step in vision transformers (Dosovitskiy et al., 2020)). To handle variable neuron counts across sessions and animals, NDT2 uses a session-specific linear *stitcher* layer that projects each session's spike bins into a shared hidden dimension before the encoder and context embeddings.

NDT2 is pretrained with a masked autoencoding objective across a large collection of sessions (see Figure A14 for an illustration of the masking scheme). For adaptation to a new session, the model first undergoes calibration: the model is finetuned end-to-end on unlabeled neural activity from the evaluation session using the same masked autoencoding objective. The calibrated encoder is then finetuned on the labeled downstream decoding task.

We omit NDT2 from the main comparison table and analysis, as our evaluation revealed the model to be in a failure mode: performance consistently remains at or below the single-session linear baseline across tasks (Table A37), indicating that the pretrained representations do not provide meaningful nonlinear structure beyond what a simple linear decoder already captures.

We conducted a hyperparameter exploration detailed in Appendix G.4.5.

F.2.16 NEDS

Neural Encoding and Decoding at Scale (NEDS) (Zhang et al., 2025) is a large-scale semi-supervised model for jointly learning to encode neural population activity and decode behavioral variables across many sessions and animals. A key distinguishing feature of NEDS is its multimodal tokenizer: rather than processing neural activity alone, NEDS jointly tokenizes binned spike bins and behavioral variables into a single unified sequence of tokens. Neural population activity and behavioral signals are each embedded into a shared token space and concatenated into a long multimodal token sequence that is processed by a single transformer backbone. This design allows the model to learn rich associations between neural and behavioral representations within a single forward pass.

The backbone is a transformer with Rotary Position Embeddings (RoPE) (Su et al., 2024) to encode the temporal ordering of tokens within the multimodal sequence. On the output side, a session-specific linear *stitcher* layer acts as a decoder head, projecting the transformer's latent representations back into the original neuron or behavioral variable space of each session, symmetrically to the input stitcher.

To exploit this multimodal token sequence, NEDS applies several masking schemes during pretraining: neural tokens can be masked and reconstructed from behavioral tokens (encoding), behavioral tokens can be masked and reconstructed from neural tokens (decoding), or both modalities can be jointly masked and reconstructed from the unmasked context. This multi-objective masked modeling encourages the model to learn bidirectional mappings between neural population activity and behavior, going beyond unidirectional decoding. Like NDT models, NEDS relies on bin spikes and session-specific linear *stitcher* layers on the input and output sides to project variable neuron counts into and out of a shared hidden dimension.

As shown in Table A37, NEDS performance remains close to the single-session linear baseline and falls below it on several tasks (e.g., Wheel, Reward, Choice), indicating a failure to leverage the nonlinear representational capacity of the pretrained encoder beyond what a simple linear decoder already recovers. We therefore omit NEDS from the main comparison table and analysis. Appendix G.4.6 provides more details of the hyperparameter exploration conducted.

G HYPERPARAMETER SELECTION

In this section, we detail the implementation and tuning configurations for the different models as they were used in the 3 task suites. We start by introducing how we tuned the single-session, single-task baselines. In all of the sweeps, we optimize hyperparameters using the Tree-structured Parzen Estimator (TPE) algorithm (Bergstra et al., 2011) implemented in Optuna (Akiba et al., 2019). Exact hyperparameter spaces and number of trials vary depending on the model. For a given trial, models are optimized using AdamW (Loshchilov & Hutter, 2019) with a one-cycle LR scheduler (Smith & Topin, 2018). In the search spaces, the LR scheduler's division (div) factor determines the initial learning rate and is sampled from a loguniform distribution to bias towards 1. Pretrained models are also optimized with AdamW. Following common practice (He et al., 2019; Jia et al., 2018), weight decay is applied to non-embedding, non-bias parameters only; embedding, bias, and normalization layers' scale/shift parameters are excluded.

G.1 SINGLE-SESSION BASELINES FOR TS1

We implement a set of single-session baselines for TS1 to benchmark performance without any pretraining. Each model is tuned on a single task and single session at a time in a fully supervised manner. In TS1, models must be able to map from neural activity across many units to a given decoding target. Evaluation on any task from TS1 consists of producing a prediction that conforms to the standard shape of the target: either sequence-level logits or frame-level continuous predictions. While our benchmark does enforce a specific target shape to ensure consistent and fair comparison across models, including the number of timestamps for frame-level tasks, we do not enforce a particular context window for model *input* (see Figure A11). Nonetheless, for simplicity we use a fixed 1s context window for all our baselines. We encourage exploration of a more suitable context length for each task for future works and hope the community will converge on this design choice.

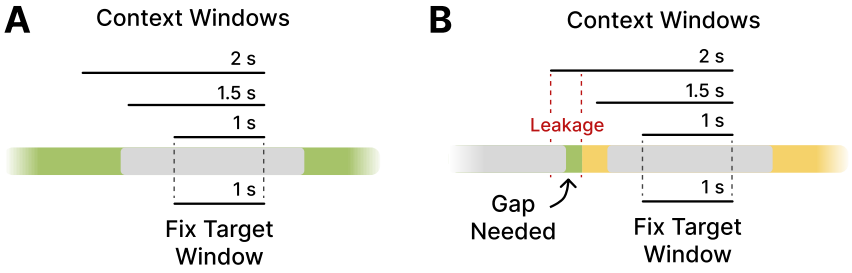


Figure A11: **Context Window Selection and Data Leakage Prevention.** (A) While our benchmark uses a fixed 1s default for both input and target windows, the framework supports variable input context lengths (e.g., 1.5s, 2s). (B) When extending the context, a temporal gap is required to prevent data leakage from outside the valid trial boundaries.

G.1.1 LINEAR

The linear baseline accepts the full input context at once, bins spike trains according to a fixed bin size, and maps directly to the target readout. We detail the tuning parameters for the linear decoding baseline in Table A11.

Table A11: Hyperparameter search space for linear decoding baselines. Linear models are tuned on individual sessions and tasks for 100 trials with 4 concurrent trials at a time.

HYPERPARAMETER	SEARCH SPACE	SAMPLING TYPE
MODEL PARAMETERS		
BIN SIZE (MS)	$\{1, 5, 10, 20, 40\}$	CATEGORICAL
TRAINING PARAMETERS		
NUMBER OF EPOCHS	$\{100, 300, 500\}$	DISCRETE (STEP 200)
BATCH SIZE	$\{16, 32, 64\}$	DISCRETE ($\log_2$)
WEIGHT DECAY	$[10^{-6}, 10^{-1}]$	LOGUNIFORM
BASE LEARNING RATE	$[10^{-5}, 10^{-2}]$	LOGUNIFORM
LR SCHEDULER – PCT START	$[0.1, 0.9]$	UNIFORM
LR SCHEDULER – DIV FACTOR	$[1, 5]$	LOGUNIFORM

G.1.2 MULTI-LAYER PERCEPTRON (MLP)

We parameterize our MLP baseline based on network depth and the width of the first hidden layer. Successive hidden layers project inputs down by a factor of 2, followed by a final readout layer. In initial experiments we found this was a reasonable design choice to balance performance enabled by expressivity of the search space with combinatorial complexity (e.g., an alternative is to allow Optuna to select a width for every layer, but the search space becomes much larger). Like the linear baseline, the MLP accepts the full context window at once and bins at a fixed bin size. We detail the tuning parameters for the MLP decoding baseline in Table A12.

Table A12: Hyperparameter search space for MLP decoding baselines. MLP's are tuned on individual sessions and tasks for 100 trials with 4 concurrent trials at a time.

HYPERPARAMETER	SEARCH SPACE	SAMPLING TYPE
MODEL PARAMETERS		
BIN SIZE (MS)	$\{5, 10, 20, 40\}$	CATEGORICAL
DEPTH	$\{1, 2, 3\}$	DISCRETE
HIDDEN DIMENSION	$\{32, 64, 128, 256, 512\}$	DISCRETE ($\log_2$)
DROPOUT	$\{0.0, 0.2, 0.4, 0.6\}$	DISCRETE (STEP 0.2)
BATCH NORM	{TRUE, FALSE}	CATEGORICAL
TRAINING PARAMETERS		
NUMBER OF EPOCHS	$\{100, 300, 500\}$	DISCRETE (STEP 200)
BATCH SIZE	$\{16, 32, 64\}$	DISCRETE ($\log_2$)
WEIGHT DECAY	$[10^{-6}, 10^{-1}]$	LOGUNIFORM
BASE LEARNING RATE†	$[10^{-5}, 10^{-2}]$	LOGUNIFORM
LR SCHEDULER – PCT START	$[0.1, 0.9]$	UNIFORM
LR SCHEDULER – DIV FACTOR	$[1, 5]$	LOGUNIFORM

† When batch normalization is disabled, the upper bound is lowered to 10^{-3} to mitigate training instability.

G.1.3 Convolutional Neural Network (CNN)

The CNN baseline is implemented on a binned raster of fixed bin size, and has a final adaptive average pooling layer before readout. The search space allows for identity activation when depth is 1, which degenerates to the rolling window linear readout baseline (see Appendix F.2.2). Note, however, that to enable fair comparison with other methods that receive context in a bidirectional fashion (e.g., by ingesting the full context window at a time), we utilize non-causal convolutions by applying padding on both the left and right of the kernels. The padding size is deterministic based on kernel size. As mentioned in Appendix F.2.2, we start with a typical implementation of the TCN (Lea et al., 2016) and fix the dilation factor to 1 for simplicity. We allow various bin sizes to improve the expressivity of the CNN baseline.

Note that CNN and GRU are typical baselines in various other fields of ML that we adapt to neural decoding, hence we treat bin size as a tunable hyperparameter to retain expressivity on par with the linear and MLP baselines. On the other hand, methods designed to work with neural data (e.g., CEBRA, NDT) are used with their original designs, including fixed bin sizes, to preserve their intended inductive biases.

We detail the tuning parameters for the CNN decoding baseline in Table A13.

Table A13: Hyperparameter search space for CNN decoding baselines. CNN's are tuned on individual sessions and tasks for 100 trials with 4 concurrent trials at a time.

Hyperparameter	Search Space	Sampling type
Model parameters		
Bin size (ms)	$\{1, 5, 10, 20, 40\}$	Categorical
Depth[†]	$\{1, 2, 4, 6, 8, 10\}$	Discrete
Hidden dimension	$\{32, 64, 128, 256\}$	Discrete ($\log_2$)
Kernel size	$\{3, 5, 7\}$	Discrete (step 2)
Dropout	$\{0.0, 0.2, 0.4, 0.6\}$	Discrete (step 0.2)
Batch norm	{True, False}	Categorical
Activation[‡]	{ReLU, Identity}	Categorical
Training parameters		
Number of epochs	$\{100, 300, 500\}$	Discrete (step 200)
Batch size	$\{16, 32, 64\}$	Discrete ($\log_2$)
Weight decay	$[10^{-6}, 10^{-1}]$	Loguniform
Base learning rate[§]	$[10^{-5}, 10^{-2}]$	Loguniform
LR scheduler – pct start	$[0.1, 0.9]$	Uniform
LR scheduler – div factor	$[1, 5]$	Loguniform

[†] Sampled via an index $i \in \{0, \ldots, 5\}$, mapped to depth 1 if $i = 0$ and $2i$ otherwise.
[‡] Only sampled when depth $= 1$ to allow the degenerate linear case; deeper models use ReLU.
[§] Upper bound is reduced to 10^{-3} for the licking rate task to mitigate training instability.

G.1.4 Gated Recurrent Unit (GRU)

The GRU baseline operates on a binned raster, projecting spike counts at each timestep into a fixed hidden dimension via a linear layer before processing the sequence through a multi-layer GRU. The context window is processed in strides determined by the bin size, rather than ingested all at once, reflecting the sequential nature of the recurrent architecture. Note that dropout is disabled within the GRU when depth is 1, as it is only meaningful across stacked recurrent layers. We include bidirectionality as a tunable hyperparameter: when enabled, separate forward and backward passes are concatenated, allowing the model to exploit future context symmetrically with the past, on par with the non-causal convolutions used in the CNN baseline. Adaptive average pooling is applied over the temporal dimension before the final linear readout to support both sequence- and frame-level tasks. The upper bound on hidden dimension is reduced from 512 to 256 at a bin size of 1 ms to satisfy memory constraints, and the learning rate upper bound is reduced to 10^{-3} on the licking rate task to mitigate training instability.

Note that GRU and CNN are typical baselines in various other fields of ML that we adapt to neural decoding, hence we treat bin size as a tunable hyperparameter to retain expressivity on par with the linear and MLP baselines. On the other hand, methods designed specifically for neural data (e.g., CEBRA, NDT) are used with their original designs, including fixed bin sizes, to preserve their intended inductive biases.

We detail the tuning parameters for the GRU decoding baseline in Table A14.

Table A14: Hyperparameter search space for GRU decoding baselines. GRU's are tuned on individual sessions and tasks for 100 trials with 4 concurrent trials at a time.

Hyperparameter	Search Space	Sampling type
Model parameters		
Bin size (ms)	{1, 5, 10, 20, 40}	Categorical
Depth	{1, 2, 3}	Discrete
Hidden dimension[†]	{32, 64, 128, 256, 512}	Discrete ($\log_2$)
Dropout	{0.0, 0.2, 0.4, 0.6}	Discrete (step 0.2)
Bidirectional	{True, False}	Categorical
Training parameters		
Number of epochs	{100, 300, 500}	Discrete (step 200)
Batch size	{16, 32, 64}	Discrete ($\log_2$)
Weight decay	$[10^{-6}, 10^{-1}]$	Loguniform
Base learning rate[‡]	$[10^{-5}, 10^{-2}]$	Loguniform
LR scheduler – pct start	[0.1, 0.9]	Uniform
LR scheduler – div factor	[1, 5]	Loguniform

[†] Upper bound is reduced to 256 when bin size = 1 ms to satisfy memory constraints.
[‡] Upper bound is reduced to 10^{-3} for the licking rate task to mitigate training instability.

G.1.5 CEBRA

CEBRA is applied in a two-step process. In the first step, the CEBRA encoder is fitted on the training split of each session using a contrastive objective, producing a low-dimensional embedding of binned neural activity. In the second step, a two-layer MLP probe, with hidden dimension equal to twice the CEBRA output dimension, a GELU activation, and dropout ($p = 0.2$), is trained on top of the frozen embeddings to predict the target behavior. The probe is optimized using the AdamW optimizer with a OneCycleLR scheduler, and early stopping is applied on the validation split.

Neural activity is binned at a fixed resolution of 20 ms prior to encoding and is not treated as a tunable hyperparameter following the original CEBRA paper (Schneider et al., 2023b). This stands in contrast to recurrent and convolutional baselines (e.g., GRU, CCN), which operate directly on spike trains at flexible temporal resolutions.

For sequence-level tasks, the CEBRA embeddings are mean-pooled across time before being passed to the probe, reducing the per-trial representation to a single vector. For frame-level tasks, the probe is applied at each time step independently.

The contrastive mode (`time`, `behavior`, or `hybrid`) is selected per task and per session during hyperparameter tuning. The number of hidden units in the MLP probe is coupled to the output dimension, fixed at $2\times$ the output dimension.

We detail the tuning parameters for the CEBRA decoding baseline in Table A15.

Table A15: Hyperparameter search space for CEBRA baselines. CEBRA is tuned on individual sessions and tasks for 100 trials with 4 concurrent trials at a time.

Hyperparameter	Search Space	Sampling type
Model parameters		
Output dimension	{8, 16, 32, 64, 128, 256}	Discrete ($\log_2$)
Time offsets	{1, 2, 3, 4, 5}	Discrete
Model learning rate	$[10^{-4}, 10^{-3}]$	Loguniform
Temperature	[0.5, 2.0]	Discrete (step 0.1)
Mode	{time, behavior}	Categorical
Temperature mode	{constant}	Categorical
Max iterations	{5000, 7000, 9000}	Discrete (step 2000)
Training parameters		
Number of epochs	{100, 200, 300, 400, 500}	Discrete (step 100)
Batch size	{16, 32, 64}	Discrete ($\log_2$)
Weight decay	$[10^{-6}, 10^{-1}]$	Loguniform
Base learning rate	$[10^{-5}, 10^{-2}]$	Loguniform
LR scheduler – pct start	[0.1, 0.9]	Uniform
LR scheduler – div factor	[1, 5]	Loguniform

Note: Number of hidden units is set to $2\times$ output dimension.

G.1.6 POYO

POYO uses spike tokenization (see Appendix F.1.2), where each spike is represented as a tuple (u_i, t_i) of unit identity and timestamp, making the model independent of any fixed bin size. The architecture compresses the input spike sequence into a fixed latent grid via cross-attention, processes it through several self-attention layers, and decodes predictions via a final cross-attention decoder. For sequence-level tasks, a single query token placed at the midpoint of the context window is used; for frame-level tasks, one query token is placed at each target timestamp. In both cases, the query tokens are constructed from a learnable session embedding combined with rotary position embeddings encoding the output timestamps.

Due to the memory overhead of Transformer models, POYO is tuned with 2 concurrent trials rather than the 4 used for lighter baselines. To ensure the number of Bayesian updates remains proportionally equivalent, the total trial budget is reduced to 50 accordingly, maintaining comparable hyperparameter optimization efficiency across all models.

We detail the tuning parameters for the POYO single-session baseline in Table A16.

Table A16: Hyperparameter search space for POYO decoding baselines. POYO models are tuned on individual sessions and tasks for 50 trials with 2 concurrent trials at a time.

Hyperparameter	Search Space	Sampling type
Model parameters		
Latent step	{0.0625, 0.125}	Discrete (step 0.0625)
Num. latents per step	{8, 16, 32}	Discrete ($\log_2$; sampled as 2^k, $k \in \{3, 4, 5\}$)
Depth	{2, 4, 6, 8, 10}	Discrete (step 2)
Dropout†	{0.0, 0.2, 0.4, 0.6}	Discrete (step 0.2)
Dimension	{32, 64, 128, 256}	Discrete ($\log_2$; sampled as 2^k, $k \in \{5, 6, 7, 8\}$)
Dim. head‡	{32, 64}	Conditional categorical
Num. attention heads§	{1, 2, 4, 8}	Derived as dimension / dim. head
Training parameters		
Number of epochs	{100, 300, 500}	Discrete (step 200)
Batch size	{16, 32, 64}	Discrete ($\log_2$; sampled as 2^k, $k \in \{4, 5, 6\}$)
Weight decay	$[10^{-6}, 10^{-1}]$	Loguniform
Base learning rate	$[10^{-5}, 10^{-2}]$	Loguniform
LR scheduler – pct start	[0.1, 0.9]	Uniform
LR scheduler – div factor	[1, 5]	Loguniform
Unit dropout parameters		
Min. units	{100, 200, 300, 400}	Discrete (step 100)
Max. units	{600, 800, 1000, 1200}	Discrete (step 200)
Mode units¶	$\lfloor \min + \frac{\max - \min}{3} \rfloor$	Derived

† The same sampled dropout value is applied to feed-forward, linear, and attention dropout: `ffn_dropout`, `lin_dropout`, and `atn_dropout`.
‡ `dim_head` is sampled only when dimension > 32; otherwise it is set equal to the model dimension.
§ The same derived number of heads is used for both cross-attention and self-attention: `cross_heads` = `self_heads` = `dim`/`dim_head`.
¶ The unit-dropout mode is not sampled directly; it is derived from the sampled minimum and maximum unit counts.

G.1.7 NDT SUPERVISED

NDT supervised adapts the Neural Data Transformer (Ye & Pandarinath, 2021) to a fully supervised decoding setting. Each timestep is tokenized as a population vector of binned spike counts at a fixed bin size of 20 ms matching the continuous behavior sampling rate of 50 Hz (following the same temporal tokenization scheme as the CNN and GRU; see Appendix F.1). This stands in contrast to POYO, which tokenizes individual neuron spikes as events. NDT instead operates on binned population snapshots, making the bin size an architectural choice rather than a tunable preprocessing step. Each population token is projected into a fixed-dimensional embedding via a linear layer and a learnable positional embeddings are added before the sequence is processed by a standard transformer encoder with pre-norm layers. For sequence-level tasks, the output is mean-pooled over the temporal dimension before the final linear readout, while frame-level tasks read out at every timestep as the bin size match the sampling rate.

A single unified dropout rate is applied across the feed-forward, attention, and pre- and post-encoder layers, following the ablation study in the original NDT paper (Ye & Pandarinath, 2021). Although the dataset and tasks differ from the original work, we retain this design choice to stay close to the validated configuration. An optional T-Fixup initialization scheme (Huang et al., 2020) is included in the search space, which rescales attention value weights and linear layers to stabilize training without a warm-up schedule.

Due to the memory overhead of Transformer models, NDT supervised is tuned with 2 concurrent trials rather than the 4 used for lighter baselines. To ensure the number of Bayesian updates is proportionally equivalent, the total trial budget is reduced accordingly, maintaining comparable hyperparameter optimization efficiency across all models.

We detail the tuning parameters for the NDT supervised baseline in Table A17.

Table A17: Hyperparameter search space for NDT supervised decoding baselines. NDT supervised's are tuned on individual sessions and tasks for 50 trials with 2 concurrent trials at a time.

HYPERPARAMETER	SEARCH SPACE	SAMPLING TYPE
MODEL PARAMETERS		
DIMENSION	$\{32, 64, 128, 256\}$	DISCRETE ($\log_2$)
DEPTH	$\{2, 4, 6, 8, 10\}$	DISCRETE (STEP 2)
NUM. HEADS	$\{1, 2, 4\}$	DISCRETE ($\log_2$)
FFN FACTOR	$\{1, 2, 4\}$	DISCRETE ($\log_2$)
DROPOUT	$\{0.2, 0.4, 0.6\}$	DISCRETE (STEP 0.2)
T-FIXUP INIT	{TRUE, FALSE}	CATEGORICAL
TRAINING PARAMETERS		
NUMBER OF EPOCHS	$\{100, 300, 500\}$	DISCRETE (STEP 200)
BATCH SIZE	$\{16, 32, 64\}$	DISCRETE ($\log_2$)
WEIGHT DECAY	$[10^{-6}, 10^{-1}]$	LOGUNIFORM
BASE LEARNING RATE	$[10^{-5}, 5 \cdot 10^{-2}]$	LOGUNIFORM
LR SCHEDULER – PCT START	$[0.1, 0.9]$	UNIFORM
LR SCHEDULER – DIV FACTOR	$[1, 5]$	LOGUNIFORM

Spikes are binned at 20 ms and read in through a linear projection; neither is searched. No unit dropout is applied, as the model is single-session with a fixed unit set.

G.2 Single-session Baselines for TS2

G.2.1 Statistical Baselines

The statistical baselines have no training parameters: each is fit in closed form on the training windows, and its one or two hyperparameters are selected by exhaustive grid search over the values in Table A18, scored on validation with the same Poisson metric used at test. Bin size is not tuned: all methods use the TS2 protocol's 20 ms bins, giving a 50-bin context window with a 5-bin forecast horizon and 10% of units held out for co-smoothing. The reduced-rank readouts select on an independent validation hold-out draw and are then refit for the test draw, so the scored units take no part in selection. The reduced rank is additionally capped at $\min(|O|, |H|)$, the number of observed and held-out units on the session.

Table A18: Hyperparameter search space for the statistical baselines. Each method is fit per session and task by exhaustive grid search on validation, with no gradient-based training.

Hyperparameter	Search Space	Sampling type
Population coupling		
Smoothing width σ (bins)	$\{1, 1.5, 2, 2.5\}$	Categorical
Coupling exponent γ	$\{0.5, 0.75, 1, 1.25, 1.5, 2\}$	Categorical
RRR and RRR (w/ ISI features)		
Rank k	$\{1, 2, 3, 4, 5, 8, 12, 16\}^{\dagger}$	Categorical
Ridge strength λ	$\{10^0, \ldots, 10^7\}$	Categorical ($\log_{10}$)
Trailing mean		
Window length K (bins)	$\{1, 2, 3, 5, 8, 13, 21, 34, 45\}$	Categorical
Shrinkage		
Blend weight α	$[0, 1]$	Discrete (step 0.1)
Ridge autoregression		
Window length L (bins)	$\{3, 5, 10, 20, 45\}$	Categorical
Ridge strength λ	$\{10^{-1}, \ldots, 10^4\}$	Categorical ($\log_{10}$)

† Capped at $\min(|O|, |H|)$, the number of observed and held-out units on the session.

G.2.2 AUTOENCODER (AE)

We parameterize the AE based on encoder depth, decoder depth, and the width of the first hidden layer. Successive encoder layers project the representation down by a factor of 2, while the decoder mirrors this by projecting back up by a factor of 2 at each layer, followed by a final linear readout to the full output dimensionality. As with the MLP, this symmetric bottleneck design balances expressivity and search space complexity. The AE is trained with a full reconstruction objective over all neurons and timesteps, encouraging the bottleneck to capture low-dimensional population latents, and is subsequently evaluated on co-smoothing and forecasting tasks.

We detail the tuning parameters for the AE in Table A19.

Table A19: Hyperparameter search space for Autoencoder baselines. Autoencoders are tuned on individual sessions and tasks for 100 trials with 4 concurrent trials at a time.

HYPERPARAMETER	SEARCH SPACE	SAMPLING TYPE
MODEL PARAMETERS		
BIN SIZE (MS)	$\{1, 5, 10, 20, 40\}$	CATEGORICAL
TRAINING PARAMETERS		
NUMBER OF EPOCHS	$\{100,\ 300,\ 500\}$	DISCRETE (STEP 200)
BATCH SIZE	$[16, 128]$	DISCRETE ($\log_2$)
WEIGHT DECAY	$[10^{-6}, 10^{-1}]$	LOGUNIFORM
BASE LEARNING RATE	$[10^{-5}, 10^{-2}]$	LOGUNIFORM
LR SCHEDULER - PCT START	$[0.1, 0.9]$	UNIFORM
LR SCHEDULER - DIV FACTOR	$[1, 5]$	LOGUNIFORM

† Applied identically to encoder, pre-encoder, and post-encoder layers.

G.2.3 NDT

The NDT baseline for TS2 shares the same Transformer encoder backbone described in Appendix G.1.7, but departs from the supervised variant in both its tokenization and training objective. Rather than a linear projection from neuron counts to a fixed hidden dimension, the hidden dimension here is defined as $N \times \max(\texttt{unit_emb_dim}, 1)$, where N is the number of units in the session. When $\texttt{unit_emb_dim} = 0$, spike counts are passed directly as raw floats. When $\texttt{unit_emb_dim} > 0$, a learned count embedding of that dimension is applied per unit and the resulting vectors are concatenated across neurons before being passed to the encoder. The number of attention heads is constrained to divide the hidden dimension evenly, and is only tuned when $\texttt{unit_emb_dim} = 2$, otherwise it is fixed to 1. The bin size is fixed to 20 ms matching the continuous behavior sampling rate.

The model is trained with a masked modelling objective in the spirit of BERT (Devlin et al., 2019), targeting full reconstruction of masked population activity under a Poisson negative log-likelihood loss. Masking is applied along the temporal dimension across all neurons simultaneously. The search space covers the overall mask ratio, the maximum contiguous block size, and the probability of block-structured versus random masking.

Dropout is sampled from a continuous range starting at 0.2 and applied uniformly to the pre-encoder, post-encoder, and within-encoder (attention and feed-forward) layers, following the ablation study in the original NDT paper (Ye & Pandarinath, 2021). Due to the memory overhead of transformer models, NDT is tuned with 2 concurrent trials; the total trial budget is set to 50 to maintain a number of Bayesian updates comparable to other baselines, as discussed in Appendix G.1.7.

We detail the tuning parameters for the NDT baseline in Table A20.

Table A20: Hyperparameter search space for NDT baselines. NDT is tuned on individual sessions for 50 trials with 2 concurrent trials at a time.

Hyperparameter	Search Space	Sampling type
Model parameters		
Unit emb. dim	$\{0, 1, 2\}$	Discrete
Num. attention heads[‡]	$\{1, 2\}$	Discrete ($\log_2$)
FFN factor	$\{1, 2\}$	Discrete ($\log_2$)
Num. encoder layers	$\{1, 2, 3, 4, 5, 6\}$	Discrete
Dropout[†]	$[0.2, 0.6]$	Uniform
Custom init	{True, False}	Categorical
Masking parameters		
Mask ratio	$[0.5, 0.9]$	Uniform
Max block size	$\{1, 2, \ldots, 7\}$	Discrete
Block mask prob.	$[0.0, 1.0]$	Uniform
Training parameters		
Number of epochs	$\{100, 300, 500\}$	Discrete (step 200)
Batch size	$\{16, 32, 64\}$	Discrete ($\log_2$)
Weight decay	$[10^{-6}, 10^{-1}]$	Loguniform
Base learning rate	$[10^{-5}, 5 \times 10^{-2}]$	Loguniform
LR scheduler – pct start	$[0.1, 0.9]$	Uniform
LR scheduler – div factor	$[1, 5]$	Loguniform

[†] Applied identically to pre-encoder, post-encoder, and within-encoder (attention and feed-forward) dropout.
[‡] Only sampled when unit emb. dim $= 2$; otherwise set to 1 (since the model dimension is $N \cdot \max(\text{unit emb. dim}, 1)$ and must be divisible by the number of heads).

G.2.4 Latent Factor Analysis via Dynamical Systems (LFADS)

Following AutoLFADS, the regularization strengths are searched alongside the architecture rather than fixed. The encoder, controller, and initial-condition dimensions are held at 64, and bin size is not searched: LFADS uses the TS2 protocol's 20 ms bins. The corruption parameters are task-gated, since each is only read by one task's forward pass: co-smoothing searches the coordinated dropout rates, forecasting searches the reconstruction gradient scope. We detail the tuning parameters in Table A21.

Table A21: Hyperparameter search space for LFADS baselines. LFADS is tuned on individual sessions and tasks for 50 trials with 4 concurrent trials at a time.

Hyperparameter	Search Space	Sampling type
Model parameters		
Generator dimension	$\{100, 200, 400\}$	Categorical
Factor dimension	$\{20, 40, 80\}$	Categorical
Controller output dimension	$\{1, 2, 4, 8\}$	Categorical
Dropout	$[0, 0.6]$	Discrete (step 0.05)
Coordinated dropout rate†	$[0, 0.7]$	Discrete (step 0.05)
Unit dropout rate†	$[0, 0.3]$	Discrete (step 0.05)
Forecast gradient scope‡	{tail, window}	Categorical
Training parameters		
KL scale - initial condition	$[10^{-3}, 10^{1}]$	Loguniform
KL scale - controller	$[10^{-3}, 10^{1}]$	Loguniform
L2 scale - generator	$[10^{-4}, 10^{2}]$	Loguniform
L2 scale - controller	$[10^{-4}, 10^{2}]$	Loguniform
Number of epochs	$\{100, 300, 500\}$	Discrete (step 200)
Batch size	$[16, 128]$	Discrete ($\log_2$)
Weight decay	$[10^{-6}, 10^{-1}]$	Loguniform
Base learning rate	$[10^{-5}, 10^{-2}]$	Loguniform
Lr scheduler - pct start	$[0.1, 0.9]$	Uniform
Lr scheduler - div factor	$[1, 5]$	Loguniform

† Searched on co-smoothing only. ‡ Searched on forecasting only.

G.3 Single-session Baselines for TS3

G.3.1 Inter-Spike Interval (ISI)

We implemented the inter-spike interval (ISI) baseline as a two-stage pipeline: a feature extractor that computes a fixed length ISI histogram for each unit, followed by a classification probe (logistic regression or MLP) trained on those features to predict unit-level brain regions. To compute each unit's ISI histogram, we took the unit's full spike train over the entire recording session and calculated pairwise consecutive differences to obtain the ISIs. These ISIs were then binned into a 128-bin log-spaced histogram over the range $[10^{-3}, 3]$ seconds and L1-normalized to sum to one, yielding a fixed-length ISI feature vector for each unit.

We experimented with both linearly spaced and log-spaced bins for the ISI histogram. Using linearly spaced bins over the range $[0, 3]$ seconds resulted in a heavily right skewed histogram with nearly all of the probability mass being centered in the leftmost few bins. For example, with linear binning, only 2 bins covered the 0-50 ms range, whereas log-spaced binning allocated 63 bins to the same range, providing significantly finer temporal resolution at short intervals. Note that for the log-spaced binning approach, the lower bound of the range was 10^{-3} seconds, as the logarithm of 0 is undefined. Log-spaced binning yielded substantial improvement over linear binning in downstream brain region classification, with macro F1 improvement under the linear probe (single-unit: $0.215 \rightarrow 0.262$, multi-unit: $0.296 \rightarrow 0.354$) and MLP probe (single-unit: $0.288 \rightarrow 0.385$, multi-unit: $0.431 \rightarrow 0.544$).

We evaluated the ISI features using two classification probes described in D.3: logistic regression and an MLP probe. Since the ISI features are deterministic (unlike learned embeddings), we ran the linear probe once and estimated MLP variance across 5 additional seeds (43-47) using the best hyperparameters from a seed-42 MLP tuning run.

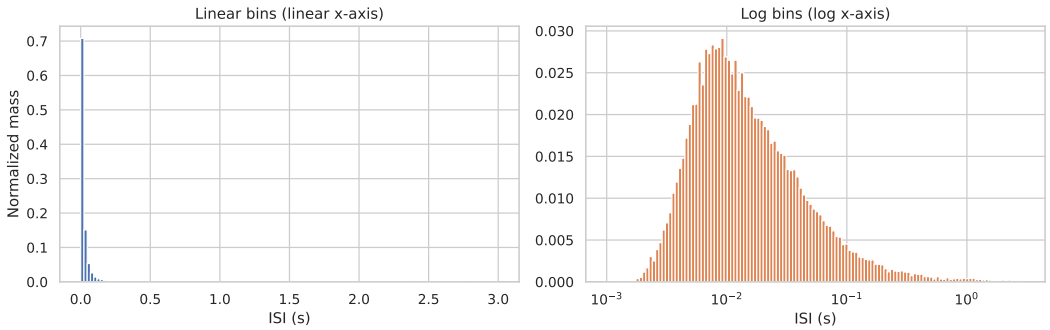


Figure A12: ISI histogram representations under linear and log-spaced binning

G.3.2 LOLCAT

We implemented the LOLCAT (Schneider et al., 2023a; Tolossa et al., 2025) baseline as per the description in Appendix F.2.7, with various implementation-specific modifications. The original LOLCAT paper divided each recording session into 3-second windows aligned with each stimulus trial (drifting grating presentations), however `BrainWideBench` does not have the same task structure. Hence, instead of using fixed-length 3-second trials, we used the task-aligned trial windows described in Figure A9 to capture the behaviorally-relevant windows, rather than arbitrarily dividing the recording session into contiguous 3-second snippets. Task-aligned trial windows are variable in length, and the number of trials per recording session varies between sessions.

As per the original LOLCAT paper, we computed an ISI histogram for each task-aligned trial window, instead of a singular ISI histogram spanning the entire recording session as we had done in the ISI baseline model (see Appendix G.3.1). This produced a variable-length set of per-trial representations per unit, which were then aggregated by the multi-head attention pooling module into a single per-unit embedding. As described in (Schneider et al., 2023a), computing per-trial histograms allows the multi-head attention module to selectively attend to the most informative trials, capturing neuron identity information at both local and global timescales. Following the same motivation as in the ISI model baseline (see Appendix G.3.1), we used log-spaced binning for the computation of the ISI histograms.

We carved out a validation set from the pretrain regime's recording sessions by using a subject-level split strategy. Recordings were grouped by subject, subjects were sorted, and the last 20% of subjects were assigned to the validation set. This subject-level split ensured that no recordings from the same subject appeared in both the training and validation sets, emulating the zero-shot conditions of the evaluation (held-out) regime.

The original architecture in the LOLCAT paper used an MLP classifier head with a single hidden layer. However, after experimenting with both linear classifiers and MLP classifiers with a hidden layer dimension of 32 and 64, we found that the linear classifier outperformed both, and thus chose to use a linear classifier head instead.

The adaptive sampler that the original LOLCAT paper used, described in Appendix F.2.7, initialized the sampling factors for each class to balance the classes (majority class count / class count). However, we found that initializing the sampling factors for each class to a uniform value of 5, rather than the class-balancing initialization, improved performance on downstream brain region classification. Additionally, we increased the multiplicative step by which the sampler adjusts each class's sampling factor at every validation epoch, from 0.99 and 1.01 to 0.96 and 1.04. With the smaller step the factors could spread by at most $1.9\times$ over a run, far short of the $19\times$ class imbalance of `BrainWideBench`, leaving the sampler unable to correct the imbalance it was designed for. The larger step reaches 10-13$\times$.

The original LOLCAT model had trial dropout (see Appendix F.2.7), but we found that it hurt brain region classification performance, hence we omitted the trial dropout from our final implementation. We experimented with setting the trial dropout probability as inversely proportional to the firing rate, as well as inversely proportional to the spike count of each trial, but both settings independently lowered performance.

Unlike the ISI baseline, no hyperparameter sweep was performed for LOLCAT; all hyperparameters were taken directly from the original LOLCAT paper (Schneider et al., 2023a), with the exception of the implementation-specific modifications mentioned above.

We detail LOLCAT training parameters in Table A22.

Table A22: LOLCAT training configuration.

HYPERPARAMETER	VALUE
MODEL PARAMETERS	
NUMBER OF ISI BINS	128
ISI RANGE (S)	$[10^{-3}, 3]$
ENCODER HIDDEN DIMS	$[128, 64, 64, 32]$
ENCODER ACTIVATION	RELU
ENCODER DROPOUT	0.5
ENCODER BATCHNORM	✓
ATTENTION HEADS	4
ATTENTION HIDDEN DIM	16
TRAINING PARAMETERS	
NUMBER OF EPOCHS	200
BATCH SIZE	64
WEIGHT DECAY	10^{-5}
BASE LEARNING RATE	10^{-3}
LR SCHEDULER – LR DECAY	10^{-2}
EARLY STOPPING PATIENCE	10
VALIDATION EVERY n EPOCHS	5
VAL FRACTION	0.2
NUMBER OF SEEDS	5
SAMPLER PARAMETERS	
INITIAL SAMPLING FACTOR	5
SAMPLING FACTOR BOUNDS	$[0.8, 100]$
GROW DIVISOR	0.96
SHRINK DIVISOR	1.04
OVERFITTING THRESHOLD	1σ
UNDERTRAINING THRESHOLD	0.1

G.3.3 NEMO

We used the NeuroPyxels Python library (Beau et al., 2021) to compute ACGs over a 2-second window (-1 second to 1 second) with 1 ms bins, using a 250 ms boxcar filter as in the original paper to estimate instantaneous firing rate. Each unit's spike train is divided into 10 quantile-based firing rate deciles (Beau et al., 2025). A separate ACG is computed for each decile and log-compressed to 201 bins (100 log-spaced positive lags mirrored symmetrically around zero), yielding a 10×201 representation per unit.

Following the original NEMO paper, each unit's waveform template was obtained by averaging approximately 500 recorded waveforms. The peak-amplitude channel was then selected from the precomputed template, and the waveform was trough-aligned to a fixed 90-sample window (42 samples before the trough, 48 after), with zero-padding applied where the trough fell near the recording edge. Each waveform was subsequently amplitude-normalized to $[-1, 1]$ by dividing by its maximum absolute value. The resulting waveforms and ACGs were stored together in a NEMO cache, from which data loaders drew samples during training. The same ACG augmentations from the original paper were retained. For waveforms, we additionally included amplitude jitter (rescaling amplitude by a uniform factor in $[0.9, 1.1]$, $p = 0.4$) alongside the Gaussian noise augmentation.

Rather than using the encoder architectures described in Section F.2.14, we scaled up both encoders to a total of approximately 10M parameters to match the scale of other models in this task suite. While scaling up, we maintained the original ratio of approximately 1.5 between the waveform and ACG encoder sizes. Each encoder was built from residual blocks consisting of two 3×3 convolutional layers with batch normalization, GeLU activations (Hendrycks & Gimpel, 2016), and dropout, with a skip connection that used a 1×1 convolution to match dimension when channel size or stride changed. The ACG and waveform encoder used 2D residual blocks and 1D residual blocks, respectively, to match the dimensionality of the ACG and waveform features.

We designed the waveform encoder as a 1D ResNet-34 style network (He et al., 2016) with a 7-sample stem convolution, followed by four stages of residual blocks with [2, 3, 3, 3] blocks. The channel widths progressed as [c, 2c, 4c, 8c], and the encoder utilized global average pooling, as well as a linear projection to the representation dimension. Similarly, we constructed the ACG encoder as a 2D ResNet-18 style network with an initial stem convolution of kernel size (3,7), followed by four stages with [2, 2, 2, 3] residual blocks and the same channel progression, with global average pooling and similar linear projection to the representation dimension.

Before finalizing the ResNet-18/ResNet-34 style encoders, we experimented with various architectures. We initially designed a simple ResNet with uniform channel widths of 256 and 128 for the waveform and ACG encoders respectively, and two residual blocks per encoder. Using the original NEMO embedding dimensions of 300 and 200, this yielded only around 1.5M parameters (868K for waveform, 618K for ACG). We then scaled these encoders up by doubling the uniform channel widths to 512 and 256, respectively, and adding a third residual block to each encoder, while scaling the embedding dimensions to 1200 and 800, reaching roughly 9.1M parameters (5.3M for waveform, 3.8M for ACG). However, we ultimately finalized the architecture as described above, since they follow established design principles of ResNet-18 and ResNet-34, such as progressive channel widening and structured block counts, rather than arbitrarily scaling channel widths or embedding dimensions.

We used early stopping based on the macro F1 score of a leave-one-session-out linear probe for brain region classification on the training set, with a patience of 250 epochs (5 validation cycles of 50 epochs each.

We detail NEMO training parameters in Table A23.

Table A23: NEMO training configuration.

HYPERPARAMETER	VALUE
MODEL PARAMETERS	
WVF BASE CHANNELS	64
ACG BASE CHANNELS	32
WVF RESIDUAL BLOCKS	$[2, 3, 3, 3]$
ACG RESIDUAL BLOCKS	$[2, 2, 2, 3]$
WVF REPRESENTATION DIM	512
ACG REPRESENTATION DIM	256
PROJECTION (EMBEDDING) DIM	512
WVF DROPOUT	0.1
ACG DROPOUT	0.2
TRAINING PARAMETERS	
NUMBER OF EPOCHS	100
BATCH SIZE	1024
WEIGHT DECAY	10^{-4}
BASE LEARNING RATE	5×10^{-4}
LR SCHEDULER – LR DECAY	10^{-2}
CONTRASTIVE LOSS TEMPERATURE	0.5
EARLY STOPPING PATIENCE	5
VALIDATION EVERY n EPOCHS	10
NUMBER OF SEEDS	5

G.4 Pretrained Baselines

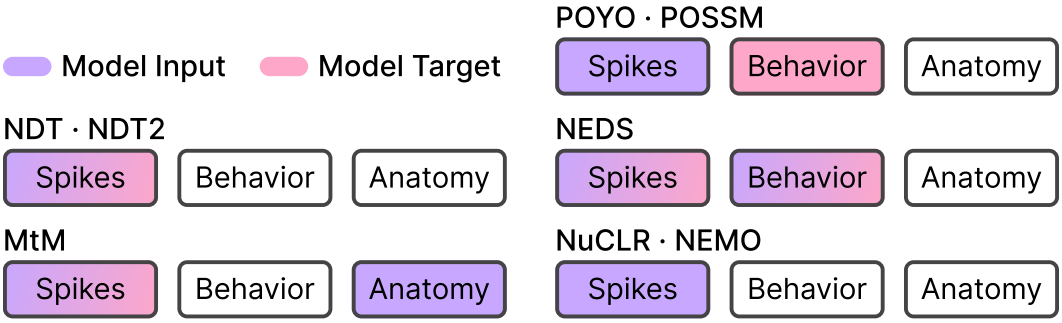


Figure A13: **Input and Target Modalities for Pretrained Baselines**. We illustrate which data types (Spikes, Behavior, Anatomy) serve as model inputs (purple) or prediction targets (pink) for each baseline model during pretraining.

We now outline pretraining and evaluation procedures for the models summarized in Figure A13. We detail any deviations in our implementations from their respective original papers, as well as tuning procedures we followed for the pretraining and finetuning of the models, if applicable. For fairness of comparison, we scaled the pretrained models to approximately 10M parameters (excluding stitcher and embedding tables) and used bf16 precision.

G.4.1 POYO+

Multi-task pretraining and task weighting. We pretrained POYO+ with a weighted multi-task objective,

$$\mathcal{L}_{\text{total}} = \sum_t \lambda_t \mathcal{L}_t. \tag{8}$$

Pretraining was conducted in an 8-task setting with the POYO+ 10M architecture for 100 epochs, using batch size 1024 and base learning rate 5×10^{-5} on the nonfiltered processed dataset. To choose the task weights, we first ran a 10-epoch pilot training run with equal weights, i.e., $\lambda_t = 1$ for all tasks. We then grouped tasks according to their observed overfitting behavior in the pilot run. In particular, we monitored whether the training loss continued to decrease while the validation loss started to increase. Tasks that showed stronger early overfitting were assigned smaller weights. For example, the choice task began to overfit around the third epoch, so we down-weighted it substantially to $\lambda_{\text{choice}} = 0.01$. In contrast, the regression tasks did not show clear overfitting within the 10-epoch pilot run, and their validation losses decreased at similar rates; therefore, we assigned them the same weight of 1.0.

The final task weights are the following:

Sequence-level Classification

λ_{choice}	$= 0.01$
λ_{reward}	$= 0.5$
$\lambda_{\text{stimulus_contrast}}$	$= 0.1$

Frame-level Regression

$\lambda_{\text{whisker_motion}}$	$= 1.0$
$\lambda_{\text{wheel_speed}}$	$= 1.0$
$\lambda_{\text{right_paw_speed}}$	$= 1.0$
$\lambda_{\text{left_paw_speed}}$	$= 1.0$
$\lambda_{\text{licking_rate}}$	$= 1.0$

Finetuning on novel sessions. We pretrained POYO+ for 35 epochs and selected the checkpoint with the highest average validation metric across all pretraining tasks as the pretrained model initialization.

We then evaluated transfer to novel sessions following the same protocol as the single-task, single-session supervised baseline models: for each novel session and each target task, we finetuned and evaluated a separate POYO+ model from the pretrained checkpoint. Finetuning used the POYO+ 10M model with batch size 32, base learning rate 5×10^{-4} and up to 300 epochs with early stopping.

During finetuning, we used a gradual unfreezing strategy. At the beginning of finetuning, we fixed the pretrained backbone and only trained the `session_embs` and `unit_embs`. The full model was then unfrozen after epoch 40. This design allows POYO+ to first adapt its session- and unit-specific representations to each novel recording session, before updating the shared pretrained backbone. This reduces early optimization instability and helps preserve the pretrained multi-task representation during adaptation to novel sessions.

Calibration and embedding extraction for TS3. TS3 evaluates unit-level representations on sessions absent from pretraining. For POYO+, no row exists for a held-out unit in the unit embedding table, so we must *calibrate*: for each held-out session we initialize from the same pretrained checkpoint and continue the pretraining objective on that session alone, creating and fitting its unit and session embeddings. Unlike the TS1 protocol above, one calibration covers all eight tasks jointly with the same weights λ_t, and its only purpose is to produce embeddings. We calibrated one model per (session, seed) using batch size 32, base learning rate 8.0×10^{-4}, `pct_start`$= 0.5$, `div_factor`$= 1.0$, weight decay 10^{-4}, bf16 precision, and 100 epochs, with the TS1 gradual unfreezing strategy unchanged (backbone frozen initially, then unfrozen after epoch 40). Within each session we defined a causal $90/10$ split of its own domain, and drew windows from the session's task-aligned intervals intersected with the union of the tasks' domains, following the pretraining setup. Checkpoints were selected on validation loss with patience 50.

A unit's representation is its row of `unit_emb`: the pretraining part of each embeddings file is read from the shared pretrained checkpoint, and the held-out part from the calibration checkpoint of that unit's session. Nothing in the objective constrains a row's magnitude, only that it be discriminative, so the two parts differ systematically in length: mean $\|x\|$ of 4.42 for pretrained rows against 1.15–1.23 for calibrated ones, a factor of roughly 3.7. Because the probes standardize using pretraining statistics, an affine map fitted on one part preserves this ratio. We therefore apply a *scale-only* normalization before probing: the held-out half is multiplied by a single scalar so both halves have the same mean vector length, leaving every direction and the held-out set's shape unchanged. The scalar uses held-out norms but no held-out labels, which the transductive setting permits since calibration has already accessed these sessions.

G.4.2 POSSM

Multi-task pretraining and task weighting. We pretrained POSSM with the same weighted multi-task objective as POYO+, i.e.,

$$\mathcal{L}_{\text{total}} = \sum_t \lambda_t \mathcal{L}_t. \tag{9}$$

Pretraining was conducted in an 8-task setting with a 10M parameter version of POSSM for 100 epochs, using batch size 256 and base learning rate 5×10^{-5} on the nonfiltered processed dataset. We use the same task weights as POYO+ to pretrain POSSM:

Sequence-level Classification

λ_{choice}	$= 0.01$
λ_{reward}	$= 0.5$
$\lambda_{\text{stimulus_contrast}}$	$= 0.1$

Frame-level Regression

$\lambda_{\text{whisker_motion}}$	$= 1.0$
$\lambda_{\text{wheel_speed}}$	$= 1.0$
$\lambda_{\text{right_paw_speed}}$	$= 1.0$
$\lambda_{\text{left_paw_speed}}$	$= 1.0$
$\lambda_{\text{licking_rate}}$	$= 1.0$

Finetuning on novel sessions. We pretrained POSSM for 100 epochs and selected the checkpoint with the highest average validation metric across all pretraining tasks as the pretrained model initialization. We then evaluated transfer to novel sessions following the same protocol as the single-task, single-session supervised baseline models: for each novel session and each target task, we finetuned and evaluated a separate POSSM model from the pretrained checkpoint. Finetuning used the 10M parameter POSSM model with batch size 32, base learning rate 5×10^{-4} and up to 300 epochs with early stopping

During finetuning, we used a gradual unfreezing strategy like POYO+. At the beginning of finetuning, we fixed the pretrained backbone and only trained the `session_embs` and `unit_embs`. The full model was then unfrozen after epoch 40. As with POYO+, his design allows POSSM to first adapt its session- and unit-specific representations to each novel recording session, before updating the shared pretrained backbone. This reduces early optimization instability and helps preserve the pretrained multi-task representation during adaptation to novel sessions.

Calibration and embedding extraction for TS3. TS3 evaluates unit-level representations on sessions absent from pretraining. For POSSM, no row exists for a held-out unit in the unit embedding table, so we must *calibrate*: for each held-out session we initialize from the same pretrained checkpoint and continue the pretraining objective on that session alone, creating and fitting its unit and session embeddings. Unlike the TS1 protocol above, one calibration covers all eight tasks jointly with the same weights λ_t, and its only purpose is to produce embeddings. We calibrated one model per (session, seed) using batch size 32, base learning rate 8.0×10^{-4}, `pct_start`$= 0.5$, `div_factor`$= 1.0$, weight decay 10^{-4}, bf16 precision, and 100 epochs, with the gradual unfreezing strategy (backbone frozen initially, then unfrozen after epoch 20; note this differs from the finetuning strategy to unfreeze at 40, which we found in initial experiments to underperform). Within each session we defined a causal $90/10$ split of its own domain, and drew windows from the session's task-aligned intervals intersected with the union of the tasks' domains, following the pretraining setup. Checkpoints were selected on validation loss with patience 50.

A unit's representation is its row of `unit_emb`: the pretraining part of each embeddings file is read from the shared pretrained checkpoint, and the held-out part from the calibration checkpoint of that unit's session. Nothing in the objective constrains a row's magnitude, only that it be discriminative, so the two parts differ systematically in length: mean $\|x\|$ of 4.67 for pretrained rows against 0.90–0.91 for calibrated ones, a factor of roughly 5.2. Because the probes standardize using pretraining statistics,

an affine map fitted on one part preserves this ratio. We therefore apply a *scale-only* normalization before probing: the held-out half is multiplied by a single scalar so both halves have the same mean vector length, leaving every direction and the held-out set's shape unchanged. The scalar uses held-out norms but no held-out labels, which the transductive setting permits since calibration has already accessed these sessions. Also note that we do not apply this scaling on linear probes as we found it to underperform compared to taking raw embedding scales, whereas with MLP probes it is necessary to prevent collapse.

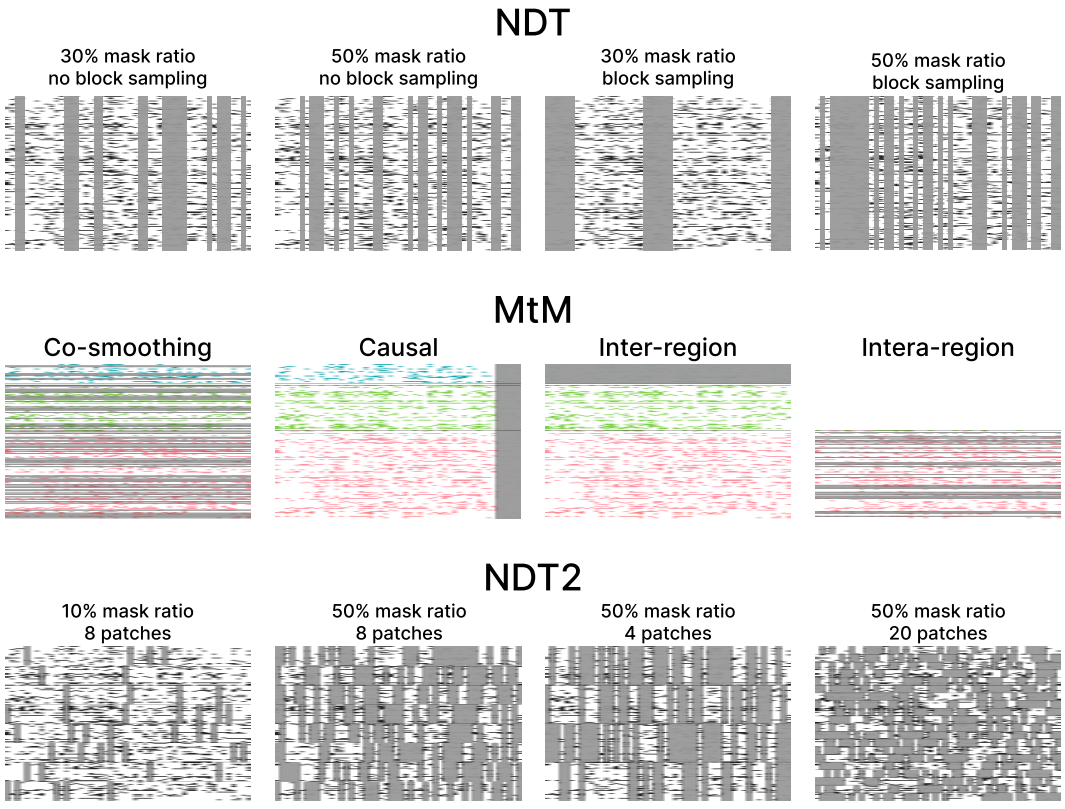


Figure A14: **Visualizing Masking Strategies for Pretrained Neural Models**. Comparison of masking schemes used during pretraining for NDT, MtM, and NDT2. Gray shaded areas represent masked spikes (targets for reconstruction).

G.4.3 NDT-STITCH

The model is pretrained with masked prediction using block masking, then fine-tuned on each task. A key finding during development was that reducing the batch size to 16 was critical to obtain good performance; larger batch sizes (32, 64, 128) consistently underperformed even after extensive learning rate sweeping across several orders of magnitude. After pretraining, we sweep the learning rate in finetuning with these values $\{10^{-5}, 2 \times 10^{-5}, 5 \times 10^{-5}, 10^{-4}, 2 \times 10^{-4}\}$.

We detail NDT-Stitch training parameters in Table A24.

Table A24: Pretraining configuration.

HYPERPARAMETER	VALUE
MODEL PARAMETERS	
BACKBONE PARAMETERS	10.42M
TOTAL PARAMETERS	43.04M
BIN SIZE (S)	0.02
HIDDEN DIMENSION	256
ENCODER NUM. LAYERS	13
ENCODER NUM. HEADS	8
ENCODER FF DIMENSION	1028
ENCODER ACTIVATION	RELU
ENCODER DROPOUT	0.2
PRE-ENCODER DROPOUT	0.2
POST-ENCODER DROPOUT	0.2
T-FIXUP SCALE BASE	0.67
T-FIXUP V SCALE FACTOR	$\sqrt{2}$
CUSTOM INITIALIZATION	✓
MASKING PARAMETERS	
MASK RATIO	0.5
MAX BLOCK SIZE	7
BLOCK MASK PROBABILITY	0.5
TRAINING PARAMETERS	
NUMBER OF EPOCHS	200
BATCH SIZE	16
WEIGHT DECAY	10^{-4}
BASE LEARNING RATE	10^{-4}
LR SCHEDULER – PCT START	0.5
LR SCHEDULER – DIV FACTOR	1

Calibration and embedding extraction for TS3. TS3 evaluates unit-level representations on sessions absent from pretraining, which for NDT-Stitch requires a stitcher for each held-out session. We therefore *calibrate*: for each held-out session we initialize the backbone from the pretrained checkpoint, initialize that session's input and output stitchers from scratch, and continue the masked-prediction objective on that session alone, with the same masking parameters used in pretraining (mask ratio 0.5, max block size 7, block mask probability 0.5). No gradual unfreezing is used: the entire model trains from the first epoch. We calibrated one model per (session, seed) using batch size 32, base learning rate 10^{-4} (one-cycle peak $\eta_{\max} = \eta_{\text{base}}\sqrt{B} = 5.7 \times 10^{-4}$), `pct_start`$= 0.5$, `div_factor`$= 1.0$, weight decay 10^{-4}, bf16 precision, and up to 500 epochs. Within each session we defined a causal $90/10$ split of its own domain, and drew windows from the session's task-aligned intervals, following the pretraining setup. Checkpoints were selected on validation loss with patience 50.

A unit's representation is the concatenation of its column of that session's input stitcher and its row of the output stitcher. Both the pretraining and held-out parts of each embeddings file are read this way, from the pretrained checkpoint and from each session's calibration checkpoint respectively. Because a stitcher must map its session's spike counts onto the scale the backbone expects, the objective pins its magnitude per session, and the two parts agree in length without intervention (mean $\|x\|$ of 1.44 against 1.47), hence no normalization is required on the embedding scales.

G.4.4 MtM

MtM shares a very similar architecture to NDT-Stitch, consistent with NDT-Stitch, we found batch size to be critical: reducing it to 16 was necessary to observe stable transfer to downstream tasks, and larger batch sizes failed to match this performance even after extensive learning rate sweeping. We applied the same fine-tuning strategy as NDT-Stitch.

We detail MtM training parameters in Table A25.

Table A25: Model and Training Configuration.

Hyperparameter	Value
Model parameters	
Bin size (s)	0.02
Hidden dimension	256
Encoder num. layers	12
Encoder num. heads	8
Encoder FF dimension	1028
Encoder activation	GELU
Encoder dropout	0.2
Pre-encoder dropout	0.2
Post-encoder dropout	0.2
T-Fixup scale base	0.67
T-Fixup V scale factor	$\sqrt{2}$
Custom initialization	✓
Masking parameters	
Mask ratio	0.3
Num. mask regions	1
Causal ratio	0.1
Training parameters	
Number of epochs	200
Batch size	16
Weight decay	10^{-4}
Base learning rate	10^{-4}
LR scheduler – pct start	0.15
LR scheduler – div factor	10
Precision	bf16

Calibration and embedding extraction for TS3. TS3 evaluates unit-level representations on sessions absent from pretraining, which for MtM requires a stitcher for each held-out session. We therefore *calibrate*: for each held-out session we initialize the backbone from the pretrained checkpoint, initialize that session's input and output stitchers from scratch, and continue the masked-prediction objective on that session alone. Following the finetuning recipe rather than the pretraining configuration above, calibration uses a harder mask (mask ratio 0.5, causal ratio 0.5, sampling over neuron, causal, inter-region and intra-region mask types) and a flat one-cycle schedule. No gradual unfreezing is used: the entire model trains from the first epoch. We calibrated one model per (session, seed) using batch size 32, base learning rate 10^{-4} (one-cycle peak $\eta_{\max} = \eta_{\text{base}}\sqrt{B} = 5.7 \times 10^{-4}$), `pct_start`$= 0.5$, `div_factor`$= 1.0$, weight decay 10^{-4}, bf16 precision, and up to 500 epochs. Within each session we defined a causal $90/10$ split of its own domain, and drew windows from the session's task-aligned intervals, following the pretraining setup. Checkpoints were selected on validation loss with patience 50.

A unit's representation is the concatenation of its column of that session's input stitcher and its row of the output stitcher. Both the pretraining and held-out parts of each embeddings file are read this way, from the pretrained checkpoint and from each session's calibration checkpoint respectively. Because a stitcher must map its session's spike counts onto the scale the backbone expects, the objective pins its magnitude per session, and the two parts remain within a small factor of each other (mean $\|x\|$ of 1.70 against 2.22, the held-out part being the longer), hence no normalization is required on the embedding scales.

G.4.5 NDT2

A key source of difficulty in applying NDT2 to `BrainWideBench` is the mismatch between the data regime for which it was originally developed and our evaluation setting. NDT2 was designed and validated on monkey recordings from Blackrock Utah arrays, which typically yield a large number of units with relatively dense, structured spiking activity. Our benchmark, by contrast, includes Neuropixel recordings from mouse, which differ substantially in array geometry, unit count, and spike statistics. This introduces a fundamental challenge around patch size: the number of neurons per patch directly determines the sequence length seen by the encoder, and hence GPU memory requirements. When units are filtered with QC, the reduced neuron count allows smaller, more tractable patches. However, our evaluation protocol requires using *unfiltered* units, which inflates the neuron count and forces larger patches to remain within memory budgets. This introduces a systematic gap between the pretraining regime, where filtered units and smaller patches can be used, and evaluation, where larger patches are imposed by the unfiltered setting, potentially degrading the quality of learned representations.

Despite these challenges, pretraining appeared to proceed successfully: reconstruction quality on held-out masked regions, measured via the D^2 metric from TS2, was positive and indicative of meaningful learned structure. However, this did not translate to downstream decoding performance. We attempted both the original paper's calibration protocol (SSL finetuning on the target session followed by supervised decoding) and direct full finetuning, sweeping learning rates across both stages, but the best results obtained are those reported in Table A37, which remain at or below the linear baseline.

We omit NDT2 from the main comparison table and analysis, as our evaluation revealed the model to be in a failure mode: performance consistently remains at or below the single-session linear baseline across tasks, indicating that the pretrained representations do not provide meaningful nonlinear structure beyond what a simple linear decoder already captures. We note that this outcome also highlights a broader difficulty in iterating on methods of this class: because results depend jointly on pretraining scale, patch size, filtering choices, and downstream finetuning, the exploration space is large and each candidate configuration requires rerunning a full pretraining pipeline, severely limiting the number of configurations that can be practically evaluated.

Full details of the hyperparameter search, including learning rate schedules, calibration duration, stitcher initialization strategies, patch size configurations, and finetuning depth (frozen encoder vs. full finetuning), are reported in Appendix G.4.5.

G.4.6 NEDS

Notably, NEDS shares important similarities with our benchmark setting: the original model was also pretrained on data from the International Brain Laboratory (IBL et al., 2021), providing a degree of overlap with our pretraining distribution. However, key differences remain. The original NEDS pretraining operates at a smaller scale, spanning up to 83 animals, and uses a longer context window of approximately 2 seconds centered around stimulus onset. Furthermore, the original pretraining covers a narrower set of task types 2 sequence-level and 2 frame-level targets — which may limit transfer to the broader range of decoding tasks evaluated in our benchmark. An additional source of discrepancy is our evaluation protocol, which uses causal temporal splits (TS1) that differ from the original training and evaluation setup.

We were unable to reproduce the original pretrained model results. We re-implemented it in the benchmark codebase and pretrained it from scratch. Pretraining proceeded successfully, even if we observed that some tasks can be overfit more easily than others (sequence-level tasks in particular). However, finetuning on downstream evaluation sessions did not yield strong decoding performance: we observed a consistent generalization gap between validation and test sets, mirroring the failure mode encountered with NDT2. We swept learning rates across finetuning stages and additionally explored a calibration strategy analogous to NDT2's neural calibration, using the masked pretraining objective to adapt the model to each new evaluation session before supervised finetuning. This calibration did not improve downstream performance, and the best results obtained across all configurations are those reported in Table A37.

H COMPUTING RESOURCES

Table A26: Computational resources used across the three task suites. Hyperparameter tuning sweeps are run on A10 GPUs (AMD EPYC 7502); all other stages are run on B200 GPUs (Xeon Platinum 8570). Tuning runtimes are estimated by sampling 200 runs; all other figures are measured from W&B run logs for the released projects, excluding crashed and failed runs. Training/finetuning runtimes are reported as the mean over all runs in the block, where a block spans all tasks × sessions × seeds.

Suite	Model	Stage	GPU	Trials/Seeds	Runtime	Total (GPU-hr)
TS1	Linear	Tuning	A10	100	1.2 hr (sweep)	339.6
	MLP	Tuning	A10	100	1.3 hr (sweep)	378.5
	GRU	Tuning	A10	100	1.5 hr (sweep)	429.9
	TCN	Tuning	A10	100	1.3 hr (sweep)	386.1
	CEBRA	Tuning	A10	100	9.1 hr (sweep)	2638.7
	POYO	Tuning	A10	50	1.2 hr (sweep)	352.6
	NDT	Tuning	A10	50	0.9 hr (sweep)	256.7
	Linear	Training	B200	1160	2.1 min / run	41.0
	MLP	Training	B200	1160	1.8 min / run	35.7
	GRU	Training	B200	1160	2.1 min / run	40.1
	CNN	Training	B200	1160	2.1 min / run	39.7
	CEBRA	Training	B200	1160	5.2 min / run	101.2
	POYO	Training	B200	1160	2.1 min / run	41.5
	NDT	Training	B200	1160	1.9 min / run	36.2
	POYO+	Pretraining	B200	1	6.0 hr / seed	6.0
	POSSM	Pretraining	B200	1	7.8 hr / seed	7.8
	NDT-stitch	Pretraining	B200	1	43.4 hr / seed	43.4
	MtM	Pretraining	B200	1	45.8 hr / seed	45.8
	POYO+	Finetuning	B200	1160	3.8 min / run	73.2
	POSSM	Finetuning	B200	1160	1.9 min / run	36.9
	NDT-stitch	Finetuning	B200	1160	1.4 min / run	26.2
	MtM	Finetuning	B200	1160	1.6 min / run	30.6
TS2	Autoencoder	Tuning	A10	100	6.2 hr (sweep)	357.6
	NDT	Tuning	A10	50	3.2 hr (sweep)	188.1
	Autoencoder	Training	B200	290	19.8 min / run	95.8
	LFADS	Training	B200	290	29.7 min / run	143.3
	NDT	Training	B200	290	12.6 min / run	61.1
	Stat. baselines	Training	B200	290	1.1 min / run	5.3
	NDT-stitch	Finetuning	B200	290	10.4 min / run	50.4
	MtM	Finetuning	B200	290	9.9 min / run	47.7
TS3	NuCLR	Pretraining	B200	5	5.8 hr / seed	28.9
	NEMO	Pretraining	B200	5	0.2 hr / seed	1.0
	POYO+	Calibration	B200	145	21.7 min / run	52.5
	POSSM	Calibration	B200	145	10.5 min / run	25.4
	MtM	Calibration	B200	145	17.9 min / run	43.1
	NDT-stitch	Calibration	B200	145	13.2 min / run	32.0

I ADDITIONAL RESULTS

In this section, we present a wider array of results across the 3 task suites, including per-session results, additional metrics, and additional analyses.

I.1 TS1 ADDITIONAL RESULTS: BEHAVIOR PREDICTION

I.1.1 PER-SESSION BREAKDOWN

The main text reports TS1 results (Table 1) averaged across all 29 evaluation sessions. Here we provide per-session breakdowns for both the primary metrics (R^2/D^2 for regression tasks and balanced accuracy for classification tasks; Figure A15) and supplementary metrics: Pearson's r for regression (Figure A16) and F1-score for classification (Figure A17). Average per-task rankings across all 29 sessions are additionally reported in Table A27.

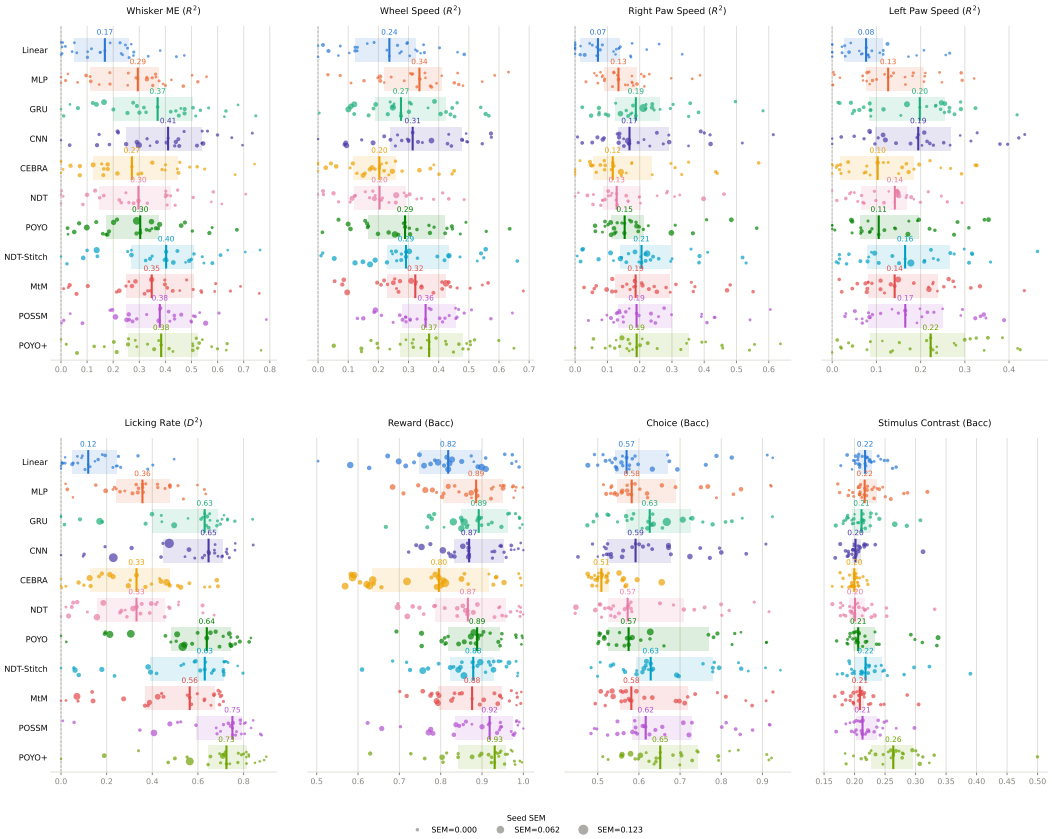


Figure A15: **Per-Session Behavior Decoding Performance Across Baselines (TS1).** Each panel shows performance for a single decoding target, Whisker Motion Energy, Wheel Speed, Right Paw Speed, and Left Paw Speed (R^2, top row), and Licking Rate (D^2), Reward, Choice, and Stimulus Contrast (balanced accuracy, bottom row), broken down by model and evaluation session. Each dot represents a single session, with dot size proportional to the standard error of the mean (SEM) across 5 random seeds; larger dots indicate higher variability across seeds. The vertical bar indicates the median performance across sessions, and the shaded region spans the interquartile range. Dots are jittered vertically for visibility.

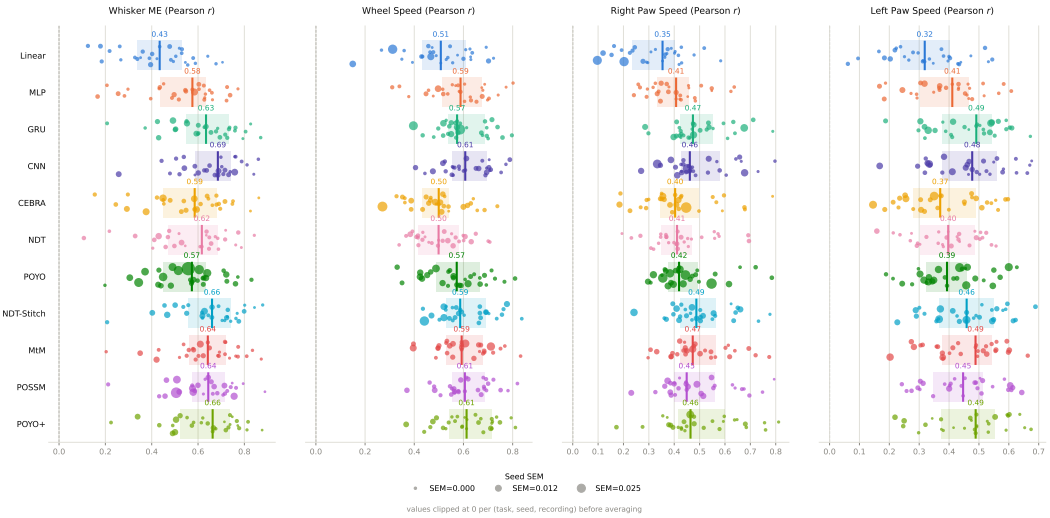


Figure A16: **Alternative Per-Session Behavior Decoding Performance Across Baselines for Regression Tasks (TS1).** Each panel shows performance for a single regression target, Whisker Motion Energy, Wheel Speed, Right Paw Speed, and Left Paw Speed, using Pearson's r, broken down by model and evaluation session. Plotting conventions as in Figure A15.

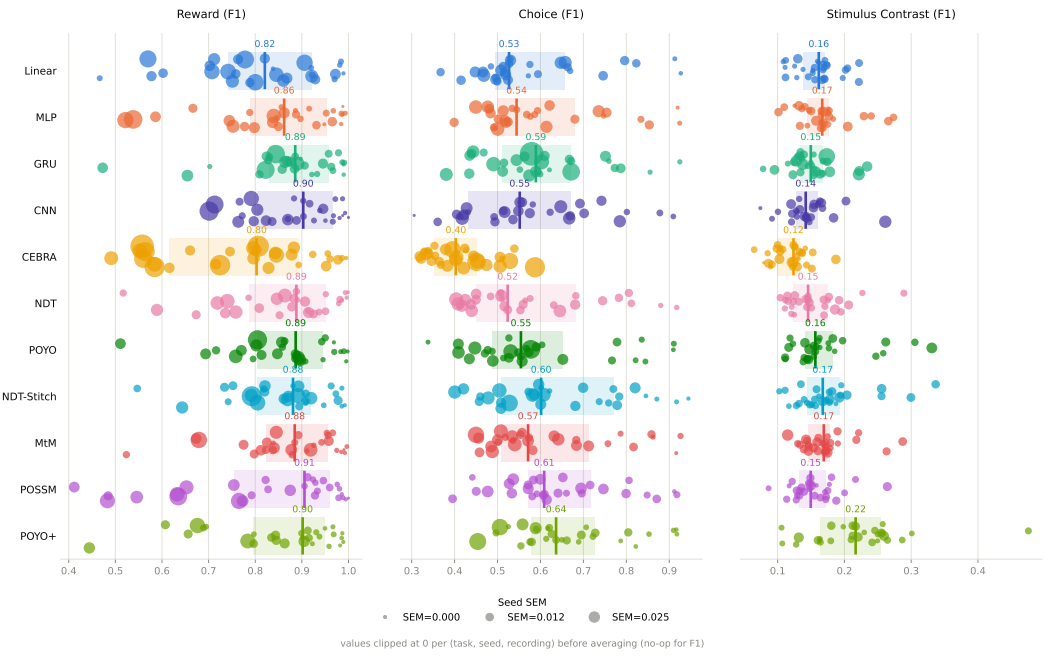


Figure A17: **Alternative Per-Session Behavior Decoding Performance Across Baselines for Classification Tasks (TS1).** Each panel shows performance for a single classification target, Reward, Choice, and Stimulus Contrast, using F1, broken down by model and evaluation session. Plotting conventions as in Figure A15.

I.1.2 PER-TASK RANKS

We report per-task average ranks in Table A27, following the procedure detailed in Appendix E.4.

Table A27: **Average per-task rankings in TS1.** Comparison of average ranks across all tasks for single-session and pretrained models. Ranks are determined by pairwise one-sided Welch's t-tests ($\alpha = 0.05$) using standard competition (1224) ranking. Lower ranks indicate better performance.

	Method	Licks	Whisker	Wheel	RPaw	LPaw	Reward	Choice	Stim	Avg. Rank ↓
Single-Session	Linear	10.10	9.66	8.52	9.10	9.38	6.31	5.07	3.69	7.73
	MLP	7.76	7.34	4.21	6.34	6.03	5.17	3.72	3.10	5.46
	GRU	4.55	3.83	4.41	4.03	3.52	3.69	4.03	3.62	3.96
	CNN	3.48	2.86	3.48	3.59	2.48	3.90	5.24	4.86	3.74
	CEBRA	8.28	7.21	8.24	6.55	7.03	6.86	8.86	6.48	7.44
	NDT	8.24	7.03	8.72	7.83	7.38	4.45	5.34	5.41	6.80
	POYO	3.52	6.45	5.00	5.31	6.03	3.66	4.07	4.34	4.80
Pretrain	POYO+	1.34	2.97	2.86	2.90	2.66	3.83	2.03	1.66	2.53
	POSSM	1.97	4.10	2.83	3.03	3.45	3.83	2.93	3.31	3.18
	NDT-Stitch	4.28	2.72	4.31	3.62	3.55	4.72	2.52	2.93	3.58
	MtM	6.10	3.59	4.14	3.86	3.31	4.17	3.17	4.07	4.05

I.1.3 SECONDARY METRICS

Table 1 reports a single primary metric per task. Here we report the secondary metrics recorded by the evaluation protocol, averaged over the 29 evaluation sessions: Pearson's r for the four continuous regression targets (Table A28), and F1-score together with average precision (AP) for the three classification targets (Table A29). These session-averaged summaries complement the per-session distributions shown in Figures A16 and A17; AP is reported here only. Licking rate admits no secondary metric under our protocol: Pearson's r is not meaningful for a count target, so Poisson D^2 is the sole metric for that task. Note that both Pearson's r and AP are insensitive to the scale of a model's outputs: r is invariant to affine rescaling of the predictions, and AP depends only on the ranking of the predicted scores. They therefore isolate how well each method tracks the target, separately from how well its outputs are calibrated. In particular, r measures co-variation with the behavioral time series independently of scale and offset, and AP measures class discriminability independently of the decision threshold.

Table A28: **Pearson's r on the TS1 regression tasks.** Session-averaged Pearson correlation between predicted and observed behavior for the four continuous targets, reported as mean ± SEM over 5 seeds. Licking rate is excluded, as Pearson's r is not appropriate for a count target. Higher is better.

	Method	Whisker	Wheel	RPaw	LPaw
Single-Session	Linear	0.431 ± 0.000	0.504 ± 0.001	0.325 ± 0.001	0.325 ± 0.001
	MLP	0.529 ± 0.001	0.583 ± 0.000	0.406 ± 0.001	0.393 ± 0.001
	GRU	0.625 ± 0.001	0.597 ± 0.002	0.499 ± 0.001	0.460 ± 0.001
	CNN	0.652 ± 0.001	0.616 ± 0.001	0.502 ± 0.003	0.472 ± 0.001
	CEBRA	0.559 ± 0.002	0.497 ± 0.000	0.420 ± 0.003	0.384 ± 0.002
	NDT	0.569 ± 0.001	0.507 ± 0.001	0.424 ± 0.001	0.392 ± 0.001
	POYO	0.558 ± 0.003	0.572 ± 0.001	0.454 ± 0.002	0.402 ± 0.002
Pretrain	POYO+	0.631 ± 0.001	0.616 ± 0.000	0.505 ± 0.002	0.468 ± 0.000
	POSSM	0.626 ± 0.002	0.613 ± 0.001	0.493 ± 0.002	0.444 ± 0.001
	NDT-Stitch	0.648 ± 0.002	0.611 ± 0.001	0.505 ± 0.003	0.462 ± 0.002
	MtM	0.629 ± 0.002	0.603 ± 0.001	0.495 ± 0.001	0.452 ± 0.000

Table A29: **Secondary metrics on the TS1 classification tasks.** Session-averaged F1-score and average precision (AP) for the three classification targets, reported as mean ± SEM over 5 seeds. Both are macro-averaged over classes. AP is not comparable across tasks, as each has a different number of classes and class prior; chance-level macro AP is approximately the mean class prevalence (0.5 for the binary tasks under a balanced prior, 0.2 for the five-way Stimulus Contrast task). Higher is better.

		Reward		Choice		Stim	
	Method	F1	AP	F1	AP	F1	AP
Single-Session	Linear	0.808 ± 0.005	0.934 ± 0.002	0.582 ± 0.005	0.675 ± 0.003	0.160 ± 0.002	0.242 ± 0.002
	MLP	0.847 ± 0.006	0.967 ± 0.001	0.600 ± 0.005	0.692 ± 0.002	0.170 ± 0.001	0.248 ± 0.002
	GRU	0.874 ± 0.005	0.961 ± 0.003	0.600 ± 0.008	0.686 ± 0.006	0.151 ± 0.002	0.242 ± 0.002
	CNN	0.883 ± 0.005	0.969 ± 0.002	0.575 ± 0.002	0.652 ± 0.002	0.145 ± 0.001	0.228 ± 0.001
	CEBRA	0.775 ± 0.010	0.898 ± 0.007	0.414 ± 0.007	0.549 ± 0.001	0.118 ± 0.002	0.223 ± 0.002
	NDT	0.857 ± 0.005	0.963 ± 0.001	0.577 ± 0.004	0.651 ± 0.002	0.154 ± 0.002	0.233 ± 0.001
	POYO	0.865 ± 0.003	0.957 ± 0.006	0.591 ± 0.008	0.665 ± 0.006	0.173 ± 0.002	0.248 ± 0.003
Pretrain	POYO+	0.850 ± 0.002	0.957 ± 0.001	0.669 ± 0.004	0.750 ± 0.003	0.218 ± 0.001	0.297 ± 0.001
	POSSM	0.821 ± 0.005	0.957 ± 0.001	0.643 ± 0.006	0.707 ± 0.004	0.155 ± 0.002	0.249 ± 0.001
	NDT-Stitch	0.858 ± 0.007	0.965 ± 0.001	0.643 ± 0.002	0.737 ± 0.003	0.176 ± 0.001	0.259 ± 0.001
	MtM	0.872 ± 0.002	0.967 ± 0.001	0.624 ± 0.003	0.692 ± 0.004	0.172 ± 0.003	0.236 ± 0.001

I.2 TS2 Additional Results: Neural Activity Prediction

I.2.1 Statistical Baselines for TS2

In Table A30 we report results on statistical baselines, described in Appendix I.2.1, along with the full results already in Table 2.

Table A30: **Task Suite 2: Co-smoothing and forecasting performance.** Model performance for both tasks is reported in terms of D^2 and bps. All metrics are reported as the average over evaluation sessions ($\pm$ SEM over 5 finetuning seeds). Rankings incorporate statistical significance, see Appendix E.4 for more details.

	Method	Co-smoothing		Forecasting		Average
		D^2	bps	D^2	bps	Rank
Single-Session	Population coupling	0.003	0.010	–	–	–
	RRR	0.112	0.304	–	–	–
	RRR (w/ ISI features)	0.128	0.328	–	–	–
	Trailing mean	–	–	0.002	0.025	–
	Shrinkage	–	–	0.077	0.161	–
	Ridge autoregression	–	–	0.108	0.253	–
	Autoencoder	0.088 ± 0.000	0.200 ± 0.001	0.037 ± 0.001	0.065 ± 0.001	4.81
	LFADS	0.176 ± 0.000	0.421 ± 0.001	0.144 ± 0.000	0.325 ± 0.001	2.34
	NDT	0.132 ± 0.001	0.304 ± 0.003	0.158 ± 0.000	0.343 ± 0.001	2.79
Pre	NDT-Stitch	0.157 ± 0.001	0.369 ± 0.002	0.177 ± 0.000	0.384 ± 0.000	1.64
	MtM	0.191 ± 0.001	0.459 ± 0.002	0.131 ± 0.001	0.288 ± 0.001	2.57

I.2.2 PER-SESSION BREAKDOWN

The main text reports TS2 results (Table 2) averaged across all 29 evaluation sessions; here we provide the corresponding per-session breakdowns (Figure A18).

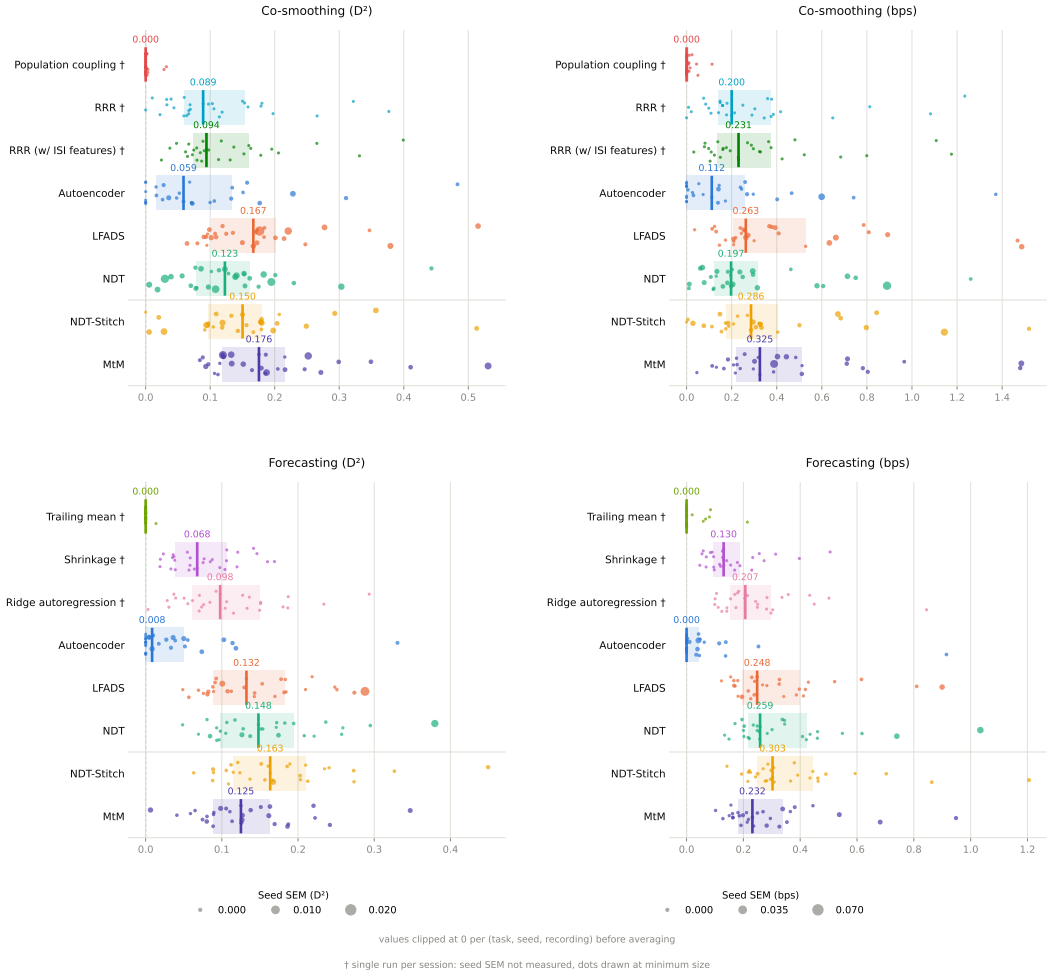


Figure A18: **Per-Session Neural Activity Prediction Performance Across Baselines (TS2).** Left panels: D^2 metric; Right panels: bps metric. Each panel shows performance for a single task (co-smoothing top row or forecasting bottom row) broken down by model and evaluation session. Plotting conventions as in Figure A15.

I.2.3 PER-TASK RANKS

In Tables A31 and A32 we provide the average per-task rankings for TS2, following the procedure detailed in Appendix E.4. Note that the stats baselines are excluded from the main rankings since they are task-specific, hence they would interfere with the rankings of the other methods that can be ranked on both tasks. Below, we include two tables with per-task rankings: one excluding the stats baselines (matching Table 2), and one including them.

Table A31: **Average per-task rankings in TS2, excluding stats baselines.** Comparison of average ranks for the co-smoothing and forecasting tasks. Ranks are determined by pairwise one-sided Welch's t-tests ($\alpha = 0.05$) using standard competition (1224) ranking. Lower ranks indicate better performance.

	Method	Co-smoothing	Forecasting	Avg. Rank ↓
SS	AE	4.69	4.93	4.81
	NDT	3.55	2.03	2.79
	LFADS	1.76	2.93	2.34
Pre.	NDT-Stitch	2.28	1.00	1.64
	MtM	1.34	3.79	2.57

Table A32: **Average per-task rankings in TS2, including stats baselines.** ComparisHon of average ranks for the co-smoothing and forecasting tasks, including stats baselines. Ranks are determined by pairwise one-sided Welch's t-tests ($\alpha = 0.05$) using standard competition (1224) ranking. Lower ranks indicate better performance.

	Method	Co-smoothing	Forecasting	Avg. Rank ↓
Single-Session	Population coupling	7.76	–	7.76
	RRR	5.14	–	5.14
	RRR (w/ ISI features)	4.62	–	4.62
	Trailing mean	–	7.59	7.59
	Shrinkage	–	5.72	5.72
	Ridge autoregression	–	5.03	5.03
	Autoencoder	6.34	6.86	6.60
	NDT	4.55	2.10	3.33
	LFADS	2.00	3.03	2.52
Pre.	NDT-Stitch	2.76	1.03	1.90
	MtM	1.34	3.93	2.64

I.3 TS3 Additional Results: Neuron Identity Prediction

Table A33 expands on the TS3 results of Table 3 by reporting per-region F1 scores across held-out sessions, offering a finer-grained view of where methods succeed and struggle across the 10 brain regions. In addition, we report per-region recall and precision in Table A34 and Table A35, respectively.

We note that in Table 3 seeds generally parameterize embedding generation, with the exception of the ISI baseline that has constant embeddings. For this we run multiple seeds on the MLP probe, though the linear probe is deterministic hence we omit SEM.

Table A33: **Brain region prediction performance across held-out sessions.** F1-scores are reported per region and as macro-average. Methods are evaluated on Single Unit (SU) and Multi-Unit (MU) data with both Linear and MLP probes. CB = Cerebellum, CNU = Caudate-Putamen, CTXsp = Cortical Subplate, HB = Hindbrain, HPF = Hippocampal Formation, HY = Hypothalamus, Iso. = Isocortex MB = Midbrain, OLF = Olfactory Areas, TH = Thalamus.

Method	Level	Probe	CB	CNU	CTXsp	HB	HPF	HY	Iso.	MB	OLF	TH	Avg
Transductive													
POYO+	SU	Linear	0.105	0.052	0.046	0.082	0.024	0.023	0.059	0.141	0.093	0.342	0.097
		MLP	0.083	0.057	0.039	0.110	0.102	0.009	0.125	0.206	0.178	0.291	0.120
	MU	Linear	0.127	0.035	0.069	0.023	0.001	0.011	0.032	0.104	0.072	0.427	0.090
		MLP	0.086	0.051	0.028	0.085	0.079	0.003	0.134	0.246	0.216	0.389	0.132
POSSM	SU	Linear	0.044	0.065	0.023	0.116	0.157	0.028	0.092	0.191	0.082	0.230	0.103
		MLP	0.089	0.086	0.050	0.131	0.177	0.031	0.137	0.183	0.167	0.182	0.123
	MU	Linear	0.005	0.027	0.019	0.113	0.215	0.021	0.079	0.261	0.024	0.273	0.104
		MLP	0.102	0.086	0.035	0.163	0.236	0.026	0.118	0.163	0.206	0.197	0.133
NDT-Stitch	SU	Linear	0.138	0.035	0.070	0.171	0.126	0.029	0.155	0.177	0.127	0.335	0.136
		MLP	0.119	0.074	0.062	0.187	0.178	0.022	0.209	0.326	0.130	0.363	0.167
	MU	Linear	0.179	0.033	0.075	0.201	0.123	0.033	0.167	0.199	0.144	0.391	0.155
		MLP	0.153	0.063	0.019	0.240	0.209	0.023	0.218	0.416	0.113	0.417	0.187
MtM	SU	Linear	0.123	0.042	0.030	0.083	0.217	0.016	0.232	0.195	0.097	0.337	0.137
		MLP	0.045	0.060	0.012	0.083	0.225	0.006	0.273	0.336	0.019	0.346	0.140
	MU	Linear	0.162	0.008	0.033	0.101	0.262	0.008	0.271	0.229	0.092	0.427	0.159
		MLP	0.012	0.059	0.008	0.064	0.290	0.000	0.308	0.409	0.003	0.429	0.158
Inductive													
ISI Baseline	SU	Linear	0.374	0.076	0.095	0.310	0.282	0.039	0.388	0.301	0.250	0.515	0.263
		MLP	0.504	0.228	0.110	0.413	0.493	0.076	0.511	0.350	0.359	0.759	0.380
	MU	Linear	0.593	0.054	0.130	0.433	0.395	0.026	0.539	0.301	0.337	0.745	0.355
		MLP	0.757	0.324	0.199	0.573	0.614	0.140	0.686	0.498	0.527	0.875	0.519
LOLCAT	SU	Linear	0.500	0.262	0.000	0.480	0.401	0.028	0.462	0.435	0.272	0.751	0.359
	MU	Linear	0.713	0.393	0.000	0.621	0.514	0.000	0.643	0.586	0.391	0.873	0.473
NEMO	SU	Linear	0.629	0.373	0.091	0.441	0.455	0.110	0.650	0.482	0.356	0.884	0.447
		MLP	0.671	0.429	0.099	0.449	0.478	0.115	0.668	0.510	0.385	0.905	0.471
	MU	Linear	0.838	0.555	0.155	0.581	0.609	0.187	0.790	0.613	0.499	0.964	0.579
		MLP	0.863	0.618	0.175	0.570	0.600	0.242	0.793	0.668	0.541	0.974	0.605
NuCLR	SU	Linear	0.883	0.200	0.241	0.841	0.635	0.222	0.739	0.867	0.595	0.934	0.616
		MLP	0.892	0.185	0.301	0.865	0.651	0.202	0.735	0.874	0.625	0.935	0.626
	MU	Linear	0.901	0.197	0.253	0.885	0.679	0.306	0.817	0.899	0.655	0.948	0.654
		MLP	0.902	0.171	0.298	0.890	0.683	0.206	0.810	0.897	0.640	0.949	0.645

Table A34: **Per-region recall on TS3 neuron identity prediction.** Recall for each of the 10 Cosmos-level brain regions, reported per method, unit level (SU: single-unit; MU: multi-unit) and probe type. Values are averaged over seeds. The rightmost column is the macro-average over regions, which for recall is exactly balanced accuracy. Higher is better.

Method	Level	Probe	CB	CNU	CTXsp	HB	HPF	HY	Iso.	MB	OLF	TH	Avg
Transductive													
POYO+	SU	Linear	0.215	0.066	0.233	0.059	0.013	0.089	0.032	0.102	0.075	0.413	0.130
		MLP	0.178	0.077	0.079	0.109	0.076	0.030	0.080	0.158	0.217	0.303	0.131
	MU	Linear	0.249	0.042	0.283	0.013	0.001	0.037	0.017	0.067	0.046	0.704	0.146
		MLP	0.165	0.061	0.050	0.076	0.051	0.007	0.087	0.203	0.227	0.526	0.145
POSSM	SU	Linear	0.036	0.057	0.033	0.197	0.194	0.130	0.056	0.146	0.065	0.251	0.117
		MLP	0.094	0.112	0.075	0.230	0.234	0.089	0.099	0.133	0.191	0.145	0.140
	MU	Linear	0.002	0.017	0.013	0.164	0.290	0.052	0.045	0.218	0.014	0.416	0.123
		MLP	0.120	0.089	0.071	0.313	0.329	0.107	0.076	0.107	0.232	0.156	0.160
NDT-Stitch	SU	Linear	0.202	0.032	0.204	0.271	0.104	0.133	0.111	0.129	0.106	0.330	0.162
		MLP	0.132	0.077	0.054	0.309	0.163	0.033	0.166	0.337	0.086	0.372	0.173
	MU	Linear	0.263	0.027	0.229	0.372	0.098	0.144	0.118	0.139	0.113	0.410	0.191
		MLP	0.153	0.053	0.013	0.426	0.178	0.019	0.175	0.464	0.069	0.460	0.201
MtM	SU	Linear	0.156	0.039	0.079	0.130	0.212	0.037	0.211	0.153	0.101	0.290	0.141
		MLP	0.026	0.056	0.008	0.091	0.225	0.004	0.311	0.434	0.011	0.319	0.148
	MU	Linear	0.191	0.006	0.083	0.172	0.257	0.015	0.267	0.176	0.097	0.373	0.164
		MLP	0.006	0.041	0.004	0.056	0.265	0.000	0.362	0.612	0.002	0.387	0.173
Inductive													
ISI Baseline	SU	Linear	0.549	0.059	0.333	0.358	0.254	0.130	0.358	0.245	0.275	0.416	0.298
		MLP	0.652	0.227	0.208	0.502	0.527	0.178	0.467	0.254	0.367	0.764	0.415
	MU	Linear	0.747	0.032	0.438	0.505	0.375	0.074	0.548	0.211	0.400	0.693	0.402
		MLP	0.835	0.286	0.271	0.730	0.685	0.200	0.680	0.359	0.553	0.946	0.554
LOLCAT	SU	Linear	0.520	0.350	0.000	0.470	0.459	0.033	0.455	0.381	0.246	0.782	0.370
	MU	Linear	0.712	0.508	0.000	0.591	0.623	0.000	0.661	0.505	0.325	0.948	0.487
NEMO	SU	Linear	0.706	0.395	0.225	0.517	0.446	0.304	0.597	0.365	0.346	0.881	0.478
		MLP	0.728	0.495	0.179	0.572	0.495	0.256	0.624	0.393	0.380	0.886	0.501
	MU	Linear	0.884	0.557	0.279	0.671	0.604	0.430	0.802	0.484	0.456	0.980	0.615
		MLP	0.891	0.675	0.204	0.672	0.624	0.389	0.812	0.554	0.520	0.981	0.632
NuCLR	SU	Linear	0.965	0.141	0.362	0.804	0.678	0.241	0.702	0.844	0.662	0.960	0.636
		MLP	0.953	0.127	0.479	0.876	0.680	0.248	0.654	0.858	0.748	0.951	0.658
	MU	Linear	0.965	0.135	0.379	0.866	0.726	0.244	0.776	0.882	0.741	0.983	0.670
		MLP	0.951	0.118	0.479	0.886	0.711	0.170	0.737	0.889	0.764	0.982	0.669

Table A35: **Per-region precision on TS3 neuron identity prediction.** Precision for each of the 10 Cosmos-level brain regions, reported per method, unit level (SU: single-unit; MU: multi-unit) and probe type. Values are averaged over seeds. The rightmost column is the macro-average over regions. Higher is better.

Method	Level	Probe	CB	CNU	CTXsp	HB	HPF	HY	Iso.	MB	OLF	TH	Avg
Transductive													
POYO+	SU	Linear	0.069	0.043	0.026	0.136	0.151	0.013	0.350	0.228	0.123	0.292	0.143
		MLP	0.055	0.049	0.027	0.114	0.156	0.006	0.305	0.299	0.154	0.286	0.145
	MU	Linear	0.086	0.031	0.039	0.096	0.033	0.007	0.468	0.244	0.175	0.307	0.149
		MLP	0.058	0.051	0.019	0.103	0.170	0.002	0.338	0.319	0.225	0.317	0.160
POSSM	SU	Linear	0.059	0.076	0.018	0.083	0.132	0.016	0.257	0.277	0.111	0.213	0.124
		MLP	0.086	0.070	0.038	0.092	0.142	0.019	0.223	0.295	0.150	0.244	0.136
	MU	Linear	0.054	0.072	0.043	0.086	0.171	0.014	0.367	0.326	0.110	0.203	0.145
		MLP	0.090	0.087	0.023	0.110	0.184	0.015	0.258	0.345	0.190	0.272	0.157
NDT-Stitch	SU	Linear	0.105	0.039	0.043	0.126	0.169	0.016	0.262	0.286	0.159	0.343	0.155
		MLP	0.114	0.073	0.074	0.134	0.200	0.017	0.282	0.316	0.265	0.357	0.183
	MU	Linear	0.137	0.046	0.046	0.139	0.203	0.019	0.296	0.361	0.218	0.383	0.185
		MLP	0.178	0.080	0.037	0.168	0.270	0.031	0.291	0.379	0.351	0.385	0.217
MtM	SU	Linear	0.102	0.048	0.019	0.061	0.222	0.011	0.257	0.273	0.096	0.405	0.149
		MLP	0.176	0.066	0.020	0.077	0.227	0.013	0.244	0.274	0.107	0.381	0.159
	MU	Linear	0.142	0.013	0.020	0.072	0.268	0.006	0.280	0.339	0.094	0.506	0.174
		MLP	0.300	0.124	0.200	0.075	0.333	0.000	0.270	0.307	0.050	0.484	0.214
Inductive													
ISI Baseline	SU	Linear	0.283	0.108	0.055	0.274	0.318	0.023	0.422	0.390	0.228	0.677	0.278
		MLP	0.416	0.239	0.076	0.352	0.463	0.049	0.565	0.565	0.353	0.755	0.383
	MU	Linear	0.492	0.171	0.076	0.379	0.417	0.016	0.529	0.527	0.290	0.806	0.370
		MLP	0.700	0.403	0.167	0.473	0.556	0.111	0.693	0.822	0.505	0.814	0.524
LOLCAT	SU	Linear	0.488	0.212	0.000	0.491	0.357	0.026	0.471	0.509	0.321	0.723	0.360
	MU	Linear	0.722	0.324	0.000	0.659	0.438	0.000	0.630	0.700	0.541	0.810	0.482
NEMO	SU	Linear	0.568	0.354	0.057	0.385	0.465	0.068	0.713	0.712	0.366	0.888	0.458
		MLP	0.623	0.379	0.068	0.370	0.463	0.075	0.719	0.725	0.390	0.924	0.474
	MU	Linear	0.798	0.553	0.107	0.512	0.614	0.120	0.779	0.838	0.550	0.948	0.582
		MLP	0.836	0.570	0.155	0.496	0.578	0.178	0.775	0.845	0.565	0.968	0.597
NuCLR	SU	Linear	0.814	0.358	0.186	0.883	0.598	0.209	0.781	0.891	0.540	0.910	0.617
		MLP	0.838	0.343	0.220	0.855	0.624	0.170	0.839	0.891	0.537	0.920	0.624
	MU	Linear	0.846	0.376	0.196	0.905	0.638	0.416	0.865	0.918	0.588	0.915	0.666
		MLP	0.859	0.319	0.217	0.895	0.658	0.262	0.901	0.905	0.550	0.917	0.648

I.4 POYO Additional Results

Table A36 compares **POYO+**, a multi-task supervised pre-trained model, against **POYO-ST**, a single-task variant that trains a separate model for each behavioral variable independently. On almost all tasks, POYO+ outperforms POYO-ST, with the largest gain on left paw (R^2: 0.200 vs. 0.132). Beyond performance, POYO-ST lacks generality: a distinct model must be trained and maintained for every task, and no shared representation is learned across behavioral variables.

POYO+, by contrast, learns a single model across all tasks simultaneously. This introduces the non-trivial challenge of *loss reweighting*: tasks with different scales, difficulties, and label frequencies must be balanced within a shared training objective, and suboptimal weighting can hurt performance on individual tasks. That POYO+ nonetheless matches or exceeds the single-task models suggests that the shared representation captures structure across behavioral variables that benefits each individual task, and further gains may be possible with improved task balancing, a known open problem in multi-task learning.

Table A36: **POYO Models Performance Summary for Frame-level Tasks.** Results are reported as in cTable 1 except avg. rank, which is not calculated. POYO+ is a multi-task supervised pre-trained model; POYO-ST is a single-task supervised pre-trained model; POYO is a single-session baseline.

Model	Licks (D^2)	Whisker (R^2)	Wheel (R^2)	RPaw (R^2)	LPaw (R^2)
POYO+	0.678 ± 0.003	0.380 ± 0.003	0.362 ± 0.001	0.249 ± 0.002	0.200 ± 0.000
POYO-ST	0.680 ± 0.006	0.319 ± 0.003	0.344 ± 0.002	0.236 ± 0.001	0.132 ± 0.003
POYO	0.559 ± 0.002	0.291 ± 0.004	0.300 ± 0.003	0.195 ± 0.002	0.136 ± 0.002

I.5 NDT2 AND NEDS ADDITIONAL RESULTS

Table A37 reports detailed results for NDT2 and NEDS alongside a single-session linear baseline. Both models perform at or below the linear baseline across most tasks: NDT2 falls short of linear on Licks and Wheel, while NEDS underperforms linear on Wheel, Reward, and Choice. Rather than capturing the nonlinear structure of neural population activity, these models largely recover what a simple linear decoder can already extract suggesting a failure to leverage the representational capacity of their architectures under the fine-tuning regime evaluated here.

Table A37: **NEDS and NDT2 Models Performance Summary.** Results are reported as in Table 1 except avg. rank, which is not calculated.

Model	Licks (D^2)	Whisker (R^2)	Wheel (R^2)	RPaw (R^2)	LPaw (R^2)	Reward (Acc)	Choice (Acc)	Contrast (Acc)
Linear	0.151 ± 0.000	0.175 ± 0.000	0.219 ± 0.002	0.091 ± 0.000	0.084 ± 0.001	0.813 ± 0.004	0.614 ± 0.004	0.219 ± 0.001
NDT2	0.127 ± 0.003	0.229 ± 0.002	0.211 ± 0.003	0.103 ± 0.004	0.100 ± 0.001	0.847 ± 0.003	0.618 ± 0.002	0.214 ± 0.001
NEDS	0.220 ± 0.012	0.243 ± 0.012	0.068 ± 0.010	0.136 ± 0.003	0.075 ± 0.003	0.659 ± 0.009	0.547 ± 0.004	0.206 ± 0.001

I.6 ADAPTING STATE-SPACE AND FLOW-FIELD METHODS

We attempted to adapt two methods that learn latent dynamics from neural population activity: CANDY (Jiang et al., 2026), a contrastive state-space model, and MARBLE (Gosztolai et al., 2025), a geometric deep learning method over local flow fields. We note that neither was particularly designed to produce general-purpose representations that transfer to held-out subjects across multi-region recordings, rather their benefits lie in their ability to produce interpretable latent dynamics that can be used for post-hoc analysis of given datasets (see Appendix J.4 for more details). Therefore, both required nontrivial modification to fit our protocol. We report the best configuration we found for each, and note that these results should be read as evidence of an adaptation gap, not as the ceiling of either approach. We encourage future works to explore the possibility of adapting these techniques more carefully to produce general-purpose representations that can transfer across animals and downstream tasks.

I.6.1 CANDY

CANDY was originally evaluated in a cross-session setting, though under a relatively limited data pool (for spikes, $\sim$40 training sessions with transfer to 2 new sessions), and on recordings from only 1–2 regions. We pretrain it with its behaviorally anchored contrastive loss together with k-step MSE forecasting. The contrastive loss is anchored jointly to all eight TS1 targets, so that a single checkpoint can serve every task suite. Discrete targets are broadcast over the context window as ordinal values (raw values for stimulus contrast).

Behavior decoding (TS1). CANDY unifies sessions differently from the stitcher/unit-embedding paradigm used by our other baselines. It learns a small shared latent dynamical system ($<$0.1% of parameters), and session-specific encoders and decoders are aligned to it through a contrastive objective. This alignment requires simultaneous access to multiple sessions. Our protocol instead adapts to one held-out session at a time. We first initialized a new session's encoder and decoder from scratch and adapted them with the forecasting objective (optionally adding the behavioral objective), but this failed to transfer. Initializing these layers with weights averaged across pretraining sessions produced nontrivial transfer (Table A38), but performance remains below baselines in the main panel, including Linear. A single-session model trained from scratch is lower on all continuous targets. On the discrete targets, both variants are near chance. Because CANDY's objective is inherently multi-session, we include the single-session variant only as a reference point, but note that it is not representative of the method.

Neural prediction (TS2). On TS2, both metrics were negative over evaluation sessions. To localize this failure mode, we evaluated each model on held-out data from its pretraining sessions, which is an in-distribution, within-session setting. On TS1, in-distribution decoding was reasonable, so the TS1 gap is attributable to transfer. On TS2, performance was below the null baseline even in-distribution, which points to a failure during pretraining itself. A plausible cause is that the shared dynamical

system must span many brain regions in our setting, whereas CANDY was developed on single-region recordings.

We also tested an auxiliary multi-task behavioral loss during pretraining, anchoring the contrastive loss to the four continuous targets only, larger latent dimensions, and several finetuning strategies. Table A38 reports the best of these.

Table A38: CANDY on TS1, finetuned on the 29 eval sessions (mean ± SEM over 3 seeds). *Pretrained*: encoder/decoder initialized from weights averaged across pretraining sessions. *Single-session*: trained from scratch on the eval session.

Target	Metric	Pretrained	Single-session
`licking_rate`	D^2	0.068 ± 0.005	0.014 ± 0.003
`whisker_motion_energy`	R^2	0.113 ± 0.002	0.019 ± 0.004
`wheel_speed`	R^2	0.113 ± 0.000	0.005 ± 0.005
`right_paw_speed`	R^2	0.055 ± 0.002	0.005 ± 0.003
`left_paw_speed`	R^2	0.051 ± 0.000	0.003 ± 0.002
`choice`	Acc	0.512 ± 0.001	0.517 ± 0.002
`reward`	Acc	0.759 ± 0.014	0.751 ± 0.004
`stimulus_contrast`	Acc	0.202 ± 0.003	0.200 ± 0.003

I.6.2 MARBLE

We implemented MARBLE as a graph-based self-supervised baseline in two stages. First, its multi-session encoder is pretrained jointly on all pretraining recordings. Then, local calibration is done on each held-out session. This stage is transductive. In particular, we: (1) fit a PCA projection on the session's neural activity, (2) build a single graph over the projected state vectors from the train, validation, and test time bins, (3) run local SSL on this graph, initialized from the pretrained encoder, and, finally, (4) freeze the resulting encoder and train a supervised readout (20 epochs) on the labeled train nodes for each behavioral target. We compare three settings: local SSL from scratch, local SSL initialized from the pretrained checkpoint (15 epochs), and a longer local adaptation (100 epochs) that follows MARBLE's original local SSL configuration as closely as possible, including learned diffusion, all-neighbor local geometry, and SSL-based validation.

Table A39 shows the results. Multi-session pretraining improves over local SSL from scratch on all four continuous targets, and longer local adaptation improves performance further. Even so, performance remains below baselines in the main panel on matched targets, including Linear. This indicates a substantial gap between MARBLE adapted to this setting and models designed for cross-session transfer at scale.

Table A39: MARBLE on TS1 continuous targets across the 29 eval sessions (R^2, 1 seed). All settings use a frozen encoder after local SSL and a 20-epoch supervised readout.

Setting	`whisker_ME`	`wheel_speed`	`r_paw_speed`	`l_paw_speed`
Local SSL from scratch (15 ep.)	0.028	−0.005	0.017	0.015
Pretrained + local SSL (15 ep.)	0.074	0.061	0.059	0.046
Pretrained + local SSL (100 ep.)	0.081	0.110	0.074	0.069

J Related Work

J.1 Neural Data Benchmarks

A growing number of benchmarks have been proposed to evaluate models of neural activity, each targeting specific aspects of neural representation learning. The Neural Latents Benchmark (NLB) (Pei et al., 2021) provides a standardized framework for evaluating latent variable models on neural population dynamics, focusing on within-session reconstruction (co-smoothing) and behavior prediction tasks. While NLB has been instrumental in advancing dynamical systems modeling, the evaluation is limited to single sessions and they do not evaluate generalization across animals or tasks.

More recent efforts have begun to incorporate cross-session and multi-task evaluation. The FALCON benchmark (Karpowicz et al., 2024) evaluates few-shot transfer for neural decoding across datasets and subjects, highlighting the importance of cross-animal generalization. However, its task structure remains relatively narrow, focusing primarily on decoding and not explicitly evaluating dynamical modeling or anatomical structure. Similarly, Sensorium (Willeke et al., 2022) evaluates large-scale neural prediction in visual cortex, enabling comparison of models for stimulus-response mapping. While Sensorium provides a strong testbed for within-domain generalization, it is restricted to a single brain region and task setting.

Beyond these, several community-driven benchmarks and competitions have focused on specific subproblems in neuroscience, including spike inference (Berens et al., 2018), spike sorting (Magland et al., 2020), and brain-computer interface decoding (Blankertz et al., 2004; Wei et al., 2022). While these efforts have been critical for advancing individual components of the analysis pipeline, they do not provide a unified evaluation of representation learning across multiple dimensions of neural data.

In contrast, `BrainWideBench` is designed to evaluate models across three complementary task families while simultaneously supporting multi-region, across-animal, and multi-task evaluation. As summarized in Table A40, this combination of properties distinguishes it from prior benchmarks, which typically focus on only one or two of these dimensions.

Table A40: **Comparison of neural data benchmarks across scale and evaluation axes.** We summarize dataset scale using reported recording hours, subjects, sessions/recordings, and recorded units. For FALCON, "Units" denotes recording channels rather than spike-sorted units. For `BrainWideBench`, we report both well-isolated units and total detected units. Prior benchmarks focus on isolated aspects of neural modeling, whereas `BrainWideBench` jointly evaluates behavior, dynamics, and anatomy across brain-wide recordings and animals.

Benchmark	Hours	Subjects	Sessions	Units	Across-Regions	Across-Subject	# Tasks	Behavior	Dynamics	Anatomy
NLB (Pei et al., 2021)	1.75 h	4	7	887	×	×	4	✓	✓	×
FALCON (Karpowicz et al., 2024)	50 h	6	68	789 ch.	×	×	5	✓	×	×
Sensorium (Willeke et al., 2022)	49 h	7	7	28k+	×	×	1	✓	×	×
Dynamic Sensorium (Turishcheva et al., 2024)	20 h	10	10	78.9k	×	×	1	✓	✓	×
`BrainWideBench` (Ours)	600+ h	139	452	611k	✓(brain-wide)	✓	3 suites, 12 tasks	✓	✓	✓

J.2 Large-scale Pretraining for Neural Data

Recent work has explored the development of large-scale pretraining for neural activity, motivated by its success in other domains (Bommasani et al., 2021; Radford et al., 2019; Achiam et al., 2023). A range of architectures have been proposed to learn representations from large neural datasets, including transformer-based approaches such as Neural Data Transformers (NDT and its variants, e.g., NDT2 and NDT-Stitch) (Ye et al., 2023), as well as models designed for multi-session and population-level learning such as POYO and POSSM (Azabou et al., 2024; Ryoo et al., 2025).

In parallel, several methods have focused on self-supervised and multimodal representation learning. Masked modeling approaches such as MtM extend ideas from language and vision to neural time series, while embedding-based methods such as NEDS (Zhang et al., 2024) and contrastive approaches such as NEMO and NuCLR (Yu et al., 2025; Arora et al., 2026) aim to align neural activity across contexts, animals, or modalities. Other works pretrain on single-neuron recordings from different acquisition modalities, most notably calcium imaging, as in POCO (Duan et al., 2026) and OmniMouse (Willeke et al., 2026). We omit these models from our comparison because applying

them to electrophysiology would require nontrivial adaptation from continuous fluorescence traces to spike trains or binned spike counts, which we leave to future work. Together, these approaches demonstrate the promise of large-scale pretraining for capturing structure in neural data, but are typically evaluated on different datasets and tasks, making it difficult to assess their relative strengths in a unified setting.

These models have demonstrated promising results in tasks such as decoding, neural prediction, and cross-session transfer. However, evaluation of these models remains fragmented, with different works focusing on different datasets, tasks, and metrics. As a result, it is difficult to assess whether improvements reflect general advances in representation learning or task-specific gains. `BrainWideBench` addresses this gap by providing a unified evaluation framework that enables systematic comparison of pretraining strategies across multiple tasks and data regimes. In doing so, it complements existing modeling work by providing a common standard for measuring progress.

J.3 Scaling and Generalization in Neural Data

A central question in the development of neural foundation models is how performance scales with data size and diversity. Inspired by scaling laws observed in language and vision models (Kaplan et al., 2020; Hoffmann et al., 2022; Cherti et al., 2023), recent work has begun to investigate scaling behavior in neural data (Antonello et al., 2023; Sato et al., 2024; Banville et al., 2025). However, emerging evidence suggests that scaling in neural data is more complex, with heterogeneity across sessions and subjects playing a critical role (Jiang et al., 2025).

At the same time, a growing body of work highlights the importance of generalization across animals and recording conditions (Safaie et al., 2023; Azabou et al., 2025). Neural activity exhibits substantial variability across individuals, making it challenging to learn invariant representations that transfer without adaptation.

By enabling evaluation across animals, regions, and tasks, `BrainWideBench` provides a platform for systematically studying scaling and generalization in representation learning from neural data. In particular, the inclusion of quality-controlled metadata allows future work to investigate how data quality, diversity, and structure impact representation learning.

J.4 Latent Dynamics and Predictive Transfer

Latent variable models and dynamical systems methods seek compact descriptions of neural population activity that reveal its underlying temporal structure. Examples include LFADS (Pandarinath et al., 2018b), which infers latent dynamics and neuron-level firing rates, CEBRA (Schneider et al., 2023b), which learns neural embeddings through contrastive learning, MARBLE (Gosztolai et al., 2025), which learns representations of local dynamical flow fields, and CANDY (Jiang et al., 2026), which learns shared, behaviorally aligned latent dynamics across recordings using contrastive alignment. Related to these approaches, NoMAD (Karpowicz et al., 2025) is a method that stabilizes behavioral decoding across days by adapting the mapping from newly recorded neural activity to a previously learned latent dynamical system while keeping its recurrent dynamics fixed. These methods support both scientific interpretation and predictive modeling: for example, the Neural Latents Benchmark evaluates latent variable models through co-smoothing, forecasting, and behavioral decoding (Pei et al., 2021). Nevertheless, predictive accuracy and faithful recovery of latent dynamics are distinct criteria: strong co-smoothing performance can coexist with extraneous latent dynamics (Dabholkar & Barak, 2026). `BrainWideBench` focuses on predictive performance and transfer across animals, sessions, and downstream tasks.

For our evaluation setting, an important consideration is how a method separates shared structure from session-specific components. Learning a common latent representation by jointly fitting several sessions does not by itself establish transfer to a session excluded from pretraining. Evaluation additionally requires an explicit procedure for adapting to newly recorded populations using the permitted target-session data. Methods that produce embeddings or flow-field descriptors may also require task-specific readouts or observation models to predict behavior or neuron-level activity. Their preprocessing and inference procedures must respect the benchmark's observation masks and temporal boundaries. These extensions require method-specific design and validation before the resulting systems can serve as comparable pretrained baselines.

The extent of parameter sharing provides a useful axis for comparing approaches. NDT-Stitch (Ye & Pandarinath, 2021) uses session-specific input and output mappings around a shared backbone, while other formulations may assign a larger encoder or decoder to each session. This changes which components benefit from pretraining and which must be estimated during adaptation; its consequences for sample efficiency and optimization require empirical evaluation. POSSM illustrates how a state-space architecture can serve as a shared computational backbone for multi-dataset pretraining and behavioral decoding (Ryoo et al., 2025). Compared to other state-space or latent dynamics models, POSSM shares more parameters across sessions during pretraining that are transferred to held-out sessions.

Our evaluation includes LFADS and CEBRA, and we additionally evaluated CANDY and MARBLE on the benchmark data. Both methods ran successfully but yielded poor predictive performance in our experiments. These findings underscore that the benefits of learning compact, aligned dynamical representations must be established under the specific predictive tasks and data regimes being evaluated. They motivate examining how representation objectives, parameter sharing, and adaptation procedures jointly affect performance on heterogeneous multi-region recordings.

J.5 Benchmarks in Other Domains

The importance of standardized benchmarks has been demonstrated across machine learning, from language understanding benchmarks such as GLUE and SuperGLUE (Wang et al., 2019) to large-scale evaluation suites for multimodal and scientific domains (Notin et al., 2023; Ren et al., 2024; Liu et al., 2024). These benchmarks have played a central role in driving progress by providing common evaluation protocols and enabling direct comparison across models.

`BrainWideBench` brings a similar philosophy to neural data, where the lack of unified benchmarks has historically made it difficult to assess progress. By integrating multiple tasks and evaluation axes into a single framework, it aims to accelerate the development of general-purpose models of neural activity.