DATA-EFFICIENT CROSS-SESSION ADAPTATION FOR BRAIN–COMPUTER INTERFACES VIA STRUCTURED DRIFT MODELING

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-term brain–computer interface (BCI) use requires neural decoders that can rapidly recover stable performance across recording sessions despite substantial neural nonstationarity. Existing cross-session adaptation methods often rely on batch-style recalibration and relatively high-dimensional parameter updates, limiting their effectiveness when only a small amount of new-session data is available. To address this limitation, we present the Distribution-based Drift Adaptation (DiDA) framework, which captures session-specific drift state through a compact distributional representation while preserving shared latent dynamics and a stable behavioral readout. By adapting this distribution rather than re-optimizing the full observation mapping, DiDA enables efficient cross-session adaptation as new trials become available sequentially. We further enable adaptation by leveraging lightweight feedback as a weak supervisory signal, rather than relying on purely unsupervised objectives. We evaluate the framework on motor cortical recordings from macaques performing center-out and random-target reaching tasks. Our experiments examine trial-by-trial adaptation under supervised, unsupervised, and preference-guided objectives, with offline batch fitting serving as a performance reference. Across these settings, DiDA enables rapid, data-efficient cross-session recalibration, highlighting its potential for long-term BCI use. Code is available at `https://anonymous.4open.science/r/DiDA-3069/`.

1 INTRODUCTION

Brain-computer interfaces (BCIs) aim to restore communication and motor function by translating neural activity into external control signals (Chaudhary et al., 2016). As BCIs move toward long-term online use, successful decoding requires high accuracy and stable performance across sessions (Pun et al., 2024). A major obstacle to this goal is the nonstationarity of neural recordings: changes in electrode impedance and recorded neural populations can alter the relationship between neural activity and intended movement over time (Sussillo et al., 2016). Consequently, decoders trained on previous sessions often degrade when applied to a new day, making repeated calibration a routine yet costly requirement for longitudinal BCI use (Card et al., 2026).

A common strategy is to learn session-specific mappings that project neural activity from different recording days into a shared representation space. Prior work has used techniques such as distribution matching (Wang et al., 2023), adversarial domain adaptation (Farshchian et al., 2019; Ma et al., 2023), and latent dynamics alignment (Vermani et al., 2024; Karpowicz et al., 2025), demonstrating that behavior-relevant neural structure can remain stable despite cross-session variability. In practical use, however, cross-session decoding is not simply a representation alignment problem. Requiring users to undergo lengthy calibration at the start of each session increases burden and can lead to fatigue, making it important to restore useful decoding performance as early as possible (Brandman et al., 2018). Consequently, the decoder should adapt as early as possible, using only the limited data available at the beginning of a new session. Most existing approaches rely on refitting high-dimensional model parameters, which typically require sufficient calibration data and are better suited to batch-style adaptation. This dependence on data-rich fitting limits their suitability for practical online use, where adaptation must be effective from sparse and streaming observations. Moreover, existing adaptation is largely model-centric: updates are driven by labels or

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.

model-internal objectives, leaving no natural mechanism to exploit lightweight user feedback when explicit supervision is unavailable. Such feedback, although coarse, can provide direct information about whether the decoder is moving toward behaviorally useful operation.

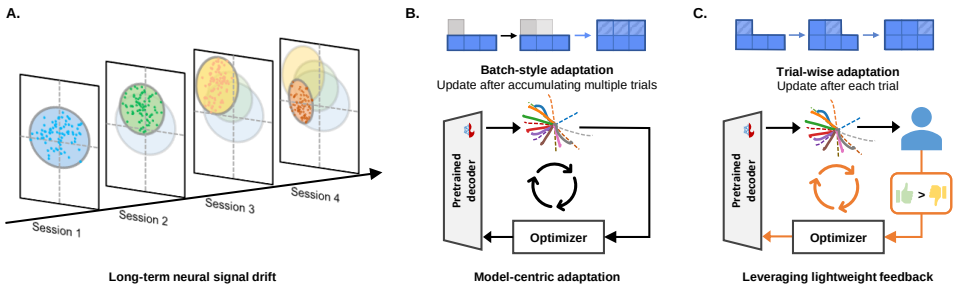

Figure 1: **Challenges in long-term BCI adaptation.** (A) Neural representations drift across recording sessions despite similar behavioral tasks, requiring cross-session recalibration. (B) Existing calibration approaches typically rely on batch-style adaptation, where model updates are performed after collecting multiple trials from a new session. Conventional adaptation objectives are often model-centric, relying on predefined representation constraints. (C) A more flexible adaptation framework can update session-specific drift online while incorporating feedback to guide recalibration.

To address this challenge, we propose a structured adaptation framework for rapid calibration. The model learns shared latent dynamics and a stable behavioral readout across historical sessions, while capturing session-specific variation through a compact drift state that modulates the observation mapping. The model reduces the number of parameters that must be inferred from new-session data, enabling calibration with only a few trials. Importantly, we model the drift state probabilistically rather than as a single point estimate, providing a distribution over plausible session-specific distortions from which candidate adaptations can be sampled. This sampling space enables preference-guided adaptation, allowing preference signals over candidate decoded outputs to refine the drift distribution without requiring precise behavioral labels. Together, these components enable streaming early-session adaptation through supervised behavioral labels, unsupervised latent-space objectives, or lightweight feedback. We evaluate the proposed framework on center-out and random-target reaching tasks under three adaptation settings: supervised adaptation with behavioral labels, label-free adaptation using unsupervised latent-space objectives, and lightweight preference feedback. Results across both settings demonstrate the effectiveness of our structured adaptation framework for rapid new-session calibration. Our main contributions are summarized as follows:

- We formulate cross-session neural decoding as a rapid, streaming adaptation problem, emphasizing early recovery of usable performance from a small prefix of new-session data rather than eventual offline accuracy.
- We propose a structured adaptation framework that confines session-specific variability to a compact drift state, enabling fast, stable, and data-efficient updates.
- We incorporate lightweight preference feedback as an alternative adaptation signal when behavioral labels are unavailable, and validate the framework on both center-out and random-target reaching tasks under supervised, unsupervised and preference-guided settings.

2 Related Work

2.1 Latent Dynamics for Neural Decoding

Neural population activity often exhibits structured temporal dynamics that can be captured in a low-dimensional latent space (Churchland et al., 2012). Latent dynamical models infer low-dimensional state evolution from noisy neural observations. They provide compact representations that preserve behaviorally relevant temporal structure for downstream decoding. LFADS introduced a sequential generative modeling framework for inferring single-trial latent dynamics and denoised firing rates from neural spiking activity (Pandarinath et al., 2018), while AutoLFADS extended this framework

to improve scalability and robustness across datasets (Keshtkaran et al., 2022). Longitudinal studies have shown that similar behaviors can be supported by stable low-dimensional population dynamics despite substantial changes in single-neuron activity (Gallego et al., 2020). This suggests that latent dynamics can serve as a robust intermediate representation for decoders, reducing sensitivity to variability in the observed neural signals. However, the mapping from recorded activity to these latent representations can itself change across recording sessions, motivating methods for cross-session neural alignment.

2.2 Cross-session Neural Alignment

Cross-session neural alignment has become a common strategy for mitigating decoder degradation caused by neural nonstationarity. These methods map session-specific activity into a shared representation so that a previously trained decoder can remain applicable. Degenhart et al. (2020) proposed a method that uses factor analysis to extract low-dimensional neural manifolds and aligns them across recording sessions via an orthogonal Procrustes transformation. Adversarial approaches align cross-session neural activity by matching session distributions, from ADAN-based domain adaptation (Farshchian et al., 2019) to CycleGAN-based mappings with improved robustness across animals and motor tasks (Ma et al., 2023). Diffusion-based approaches align neural activity across sessions by extracting the spatiotemporal structure of source-domain latent dynamics and recovering this structure in the target session through likelihood-guided alignment (Wang et al., 2023). Generative-model-based approaches leverage pretrained latent dynamical models to facilitate cross-session alignment. Vermani et al. (2024) incorporate multi-step prediction constraints into latent representation learning to capture more robust global neural dynamics. Building upon LFADS (Pandarinath et al., 2018), NoMAD (Karpowicz et al., 2025) introduces unsupervised alignment objectives to adapt new-session neural activity to a pretrained latent dynamical model without requiring behavioral labels. Existing methods have primarily addressed session variability through the design of increasingly sophisticated alignment objectives, focusing on what to optimize rather than how optimization should proceed as neural data arrive sequentially. This model-centric perspective largely overlooks the calibration process itself, particularly how to achieve effective adaptation from the earliest trials under limited data availability.

2.3 Feedback-Driven Preference Optimization

While model-centric objectives provide effective constraints for neural alignment, they typically emphasize predefined properties of neural representations. This limitation reflects a broader challenge in machine learning, where surrogate training objectives do not necessarily align with the desired quality of model outputs (Stiennon et al., 2020; Sani et al., 2024). Feedback offers a complementary source of information by providing task-relevant evaluative signals, enabling optimization to be guided by the model's actual outcomes. This principle has been extensively explored in reinforcement learning from human feedback (RLHF) (Christiano et al., 2017). Conventional RLHF approaches typically rely on a learned reward model to score generated responses, followed by policy optimization using algorithms such as proximal policy optimization (PPO) (Schulman et al., 2017; Ouyang et al., 2022). Direct preference optimization (DPO) simplifies this process by optimizing a policy directly from preference pairs without separately fitting a reward model (Rafailov et al., 2023). In BCI applications, Batzianoulis et al. (2021) employed inverse reinforcement learning to infer user preferences from EEG-based error-related potentials, enabling personalized robot behavior. Motivated by these insights, we adopt DPO to incorporate task-level feedback into neural alignment, allowing downstream performance to guide alignment beyond predefined representational objectives.

3 Methods

3.1 Problem Formulation

Building on latent dynamical approaches to neural decoding (Vermani et al., 2024; Karpowicz et al., 2025), we consider a pretrained decoder that maps neural activity to behavior through a low-

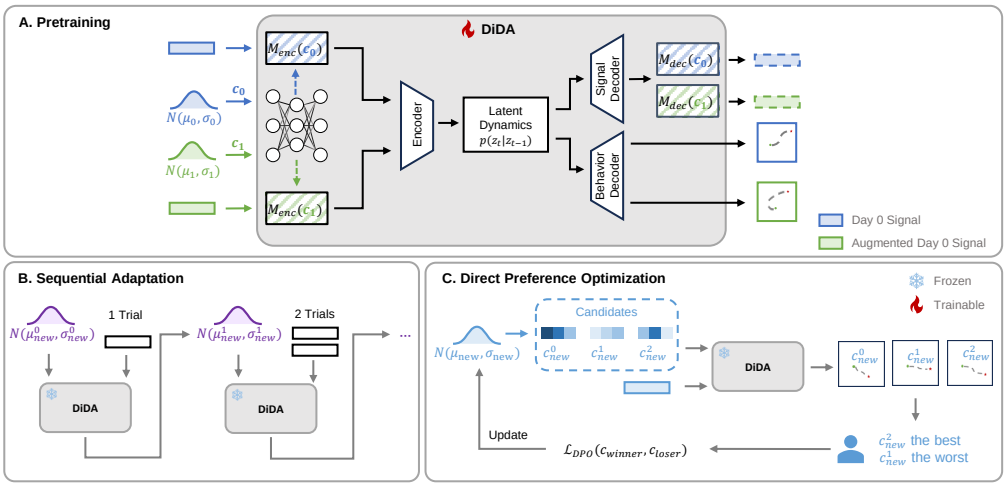

Figure 2: **Overview of DiDA framework.** (A) Pretraining: DiDA is pretrained using Day 0 neural signals and their augmented counterparts to learn shared latent dynamics and reconstruct neural signals and behavioral outputs. (B) Sequential adaptation: As trials from a new session become available, the session-specific drift distribution is progressively updated while DiDA remains frozen. (C) Direct preference optimization: Candidate drift states are sampled from the distribution. Preference feedback is obtained by identifying the best and worst decoded trajectories, which are then used to optimize the drift distribution through DPO loss.

dimensional latent space:

$$\mathbf{z}_t = f_{\phi_0}(\mathbf{x}_t), \tag{1}$$

$$\hat{\mathbf{y}}_t = g_{\psi_0}(\mathbf{z}_t), \tag{2}$$

where $\mathbf{x}_t \in \mathbb{R}^N$ denotes neural activity, $\mathbf{z}_t \in \mathbb{R}^d$ is the latent representation, and $\hat{\mathbf{y}}_t \in \mathbb{R}^D$ is the predicted behavior. Pretraining on source-session data yields a shared decoder $\Theta_0 = \{\phi_0, \psi_0\}$. When applied to a new recording session, neural drift causes a mismatch between incoming neural activity and the pretrained latent representation. We therefore formulate cross-session adaptation as updating the neural-to-latent mapping while keeping the latent dynamics and behavioral decoder fixed:

$$\phi^*_{\text{new}} = \arg\min_{\phi} \mathcal{L}_{\text{adapt}}\left(\phi; \psi_0, \mathcal{A}_{\text{new}}\right), \tag{3}$$

where $\mathcal{A}_{\text{new}}$ denotes the available adaptation information, including supervised, unsupervised, or preference-based signals.

3.2 Distribution-based Drift Parameterization

A key challenge in cross-session adaptation is the trade-off between transformation flexibility and adaptation reliability. Unconstrained transformations can capture complex session-specific changes, but their large degrees of freedom make them difficult to estimate when only limited new-session observations are available (Appendix A). We therefore parameterize session-specific changes through a probabilistic drift state. We represent neural drift in session s using a Gaussian distribution

$$q_{\eta^{(s)}}(\mathbf{c}) = \mathcal{N}\left(\boldsymbol{\mu}^{(s)}, \text{diag}\left((\boldsymbol{\sigma}^{(s)})^2\right)\right), \tag{4}$$

where $\mathbf{c} \in \mathbb{R}^m$ denotes the drift state, and $\eta^{(s)} = \{\boldsymbol{\mu}^{(s)}, \boldsymbol{\sigma}^{(s)}\}$ denotes the distribution parameters for session s. A drift state sampled from this distribution determines the session-specific neural-to-latent mapping:

$$\mathbf{c}^{(s)} \sim q_{\eta^{(s)}}(\mathbf{c}), \tag{5}$$

$$f_{\phi^{(s)}}(\mathbf{x}_t^{(s)}) = f_{\phi_0}\left(M(\mathbf{c}^{(s)})\mathbf{x}_t^{(s)}\right). \tag{6}$$

Here, $M(\mathbf{c}^{(s)})$ is the neural transformation matrix generated from the drift state by a mapping network shared across sessions. Thus, the pretrained encoder remains fixed, while the effective neural-to-latent mapping $f_{\phi^{(s)}}$ varies across sessions through $q_{\eta^{(s)}}$.

3.3 Model Pretraining

We pretrain the model using source-session data and augmented neural observations (details in Appendix B.6), treating the original and augmented data as different training sessions. For each session s, a session-specific drift state $\mathbf{c}^{(s)} \sim q_{\eta^{(s)}}(\mathbf{c})$ is sampled and used to transform the neural observations through $M(\mathbf{c}^{(s)})$. The transformed observations are then encoded by the shared encoder:

$$q_{\phi_0}\left(\mathbf{z}_t^{(s)} \mid \mathbf{x}_{1:t}^{(s)}, \mathbf{c}^{(s)}\right). \tag{7}$$

This formulation encourages the model to capture session-invariant, behaviorally relevant latent dynamics while representing session-specific variability through the drift state.

Neural variational objective. We formulate neural reconstruction and temporal regularization within a variational framework. Given a drift state $\mathbf{c}^{(s)}$, we model the neural observations through $p_\xi(\mathbf{x}_t^{(s)} \mid \mathbf{z}_t^{(s)}, \mathbf{c}^{(s)})$, while the probabilistic latent dynamics model $p_\theta(\mathbf{z}_t \mid \mathbf{z}_{t-1})$ characterizes the temporal evolution of the latent states (details in Appendix B.1). Existing sequential latent variable models typically regularize latent trajectories through local transition constraints (Pandarinath et al., 2018; Karpowicz et al., 2025). To encourage temporal consistency over longer horizons, we instead introduce a K-step-ahead dynamics objective following Vermani et al. (2024),

$$\begin{aligned}\mathcal{E}_K^{(s)} = \mathbb{E}_{q_{\eta^{(s)}}(\mathbf{c})}\mathbb{E}_{q_\phi(\mathbf{z}_{1:T}^{(s)} \mid \mathbf{x}_{1:T}^{(s)}, \mathbf{c})}\Bigg[\sum_{t=1}^{T} \log p_\xi\left(\mathbf{x}_t^{(s)} \mid \mathbf{z}_t^{(s)}, \mathbf{c}\right) \\ + \sum_{t=K+1}^{T}\left(\log p_\theta\left(\mathbf{z}_t^{(s)} \mid \mathbf{z}_{t-K}^{(s)}\right) - \log q_\phi\left(\mathbf{z}_t^{(s)} \mid \mathbf{x}_{1:t}^{(s)}, \mathbf{c}\right)\right)\Bigg].\end{aligned} \tag{8}$$

The K-step predictive density $p_\theta(\mathbf{z}_t \mid \mathbf{z}_{t-K})$ is obtained by recursively applying the one-step transition model. Since its exact evaluation requires marginalizing over intermediate latent states, we replace the predictive log-density in equation 8 with a tractable rollout-based lower bound, estimated by Monte Carlo sampling (Appendix B.2).

Behavioral decoder. We train a behavioral decoder to map the latent representation to the observed behavior. The decoder $p_\psi(\mathbf{y}_t \mid \mathbf{z}_t)$ is parameterized as a Gaussian distribution with a linear mean function and learned diagonal covariance. Its negative log-likelihood defines the behavioral loss:

$$\mathcal{L}_{\text{beh}} = -\mathbb{E}_s\left[\sum_{t=1}^{T} \log p_\psi\left(\mathbf{y}_t^{(s)} \mid \mathbf{z}_t^{(s)}\right)\right]. \tag{9}$$

This supervision encourages the latent representation to preserve behaviorally relevant information and yields a decoder that is kept fixed during cross-session adaptation.

Session variability regularization. To improve the robustness of cross-session adaptation, we regularize both the session-specific drift state and the shared latent representation during pretraining. The drift-state distributions are constrained toward a standard normal prior:

$$\mathcal{L}_{\text{drift}} = \mathbb{E}_s\left[D_{\text{KL}}\left(q_{\eta^{(s)}}(\mathbf{c}) \| p(\mathbf{c})\right)\right]. \tag{10}$$

This regularization prevents excessive deviation of session-specific drift distributions and provides a prior-based initialization for unseen sessions.

In addition, we encourage session-invariant latent representations through adversarial session classification:

$$\mathcal{L}_{\text{adv}} = \mathbb{E}_{(s,\mathbf{x})}\left[-\log h_\omega(s|\bar{\mathbf{z}})\right], \qquad \bar{\mathbf{z}} = \frac{1}{T}\sum_t \mathbf{z}_t. \tag{11}$$

The classifier is trained to predict session identity, while gradient reversal encourages the encoder to remove session-specific information from the latent space. Together, these regularizers stabilize the learned drift distribution and promote a session-robust latent representation.

Overall pretraining objective. Joint optimization learns a latent representation that captures neural dynamics and preserves behaviorally relevant information while reducing session-specific information. It also yields a behavioral decoder that can be kept fixed during cross-session adaptation and regularizes the drift-state distributions toward a shared prior for initializing new sessions.

$$\mathcal{L}_{\mathrm{pre}} = -\mathbb{E}_s\left[\mathcal{E}^{(s)}_{\mathrm{neural}}\right] + \mathcal{L}_{\mathrm{beh}} + \mathcal{L}_{\mathrm{drift}} + \mathcal{L}_{\mathrm{adv}}. \tag{12}$$

3.4 Adaptation Objectives

Supervised adaptation. Given labeled data from a new session, we optimize its drift-state distribution $q_{\eta^{(s)}}(\mathbf{c})$ while keeping the pretrained model fixed. The supervised adaptation objective is

$$\mathcal{L}_{\mathrm{sup}} = -\mathcal{E}_{\mathrm{neural}} + \mathcal{L}_{\mathrm{beh}}, \tag{13}$$

where $\mathcal{E}_{\mathrm{neural}}$ is the neural variational objective defined above, including neural reconstruction and K-step dynamics regularization, and $\mathcal{L}_{\mathrm{beh}}$ is the behavioral decoding loss.

Unsupervised adaptation. When behavioral labels are unavailable, we instead align the latent distribution of the new session with that of the source session, following NoMAD (Karpowicz et al., 2025). Let $\mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$ denote a Gaussian fitted to the pretrained source latent representations, and let $\mathcal{N}(\boldsymbol{\mu}_{\mathrm{new}}, \boldsymbol{\Sigma}_{\mathrm{new}})$ denote the corresponding Gaussian estimated from the new-session latent representations. The unsupervised objective is

$$\mathcal{L}_{\mathrm{unsup}} = D_{\mathrm{KL}}\left(\mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0) \,\|\, \mathcal{N}(\boldsymbol{\mu}_{\mathrm{new}}, \boldsymbol{\Sigma}_{\mathrm{new}})\right). \tag{14}$$

Both distributions are estimated from latent posterior means using full covariance matrices (details in Appendix B.3). Minimizing this objective aligns the new-session latent statistics with those of the source sessions without requiring behavioral labels.

Preference-guided adaptation. To enable adaptation from preference feedback, we optimize the probabilistic drift state using a DPO objective (Rafailov et al., 2023). Candidate drift states are sampled from $q_\eta(\mathbf{c})$ and mapped through the fixed model to different behavioral predictions, allowing preferences over decoded behaviors to be translated into preferences over drift states. Given a set of preference pairs $\mathcal{P} = \{(\mathbf{c}_i^+, \mathbf{c}_i^-)\}_{i=1}^{N_p}$, we optimize

$$\mathcal{L}_{\mathrm{DPO}} = -\frac{1}{N_p}\sum_{i=1}^{N_p} \log\sigma\left(\beta\left[\log\frac{q_\eta(\mathbf{c}_i^+)}{q_\eta(\mathbf{c}_i^-)} - \log\frac{q_{\mathrm{ref}}(\mathbf{c}_i^+)}{q_{\mathrm{ref}}(\mathbf{c}_i^-)}\right]\right), \tag{15}$$

where q_{ref} denotes the drift-state distribution before adaptation. This objective increases the relative likelihood of preferred drift states across multiple preference pairs, thereby shifting the drift-state distribution toward states that produce preferred behaviors (details in Appendix B.4).

4 Experiments

Datasets. We evaluate our method on publicly available long-term macaque neural recordings (Ma et al., 2023), containing motor cortical population activity and behavioral measurements collected across multiple sessions. The recordings include two reaching tasks: center-out (CO) and random-target (RT) reaching. In the CO task, the animal reaches toward one of eight predefined targets. In the RT task, the animal performs sequential reaches toward randomly located targets. Dataset and preprocessing details are provided in Appendix C.

Baselines. We compare against three cross-session alignment baselines, covering different parameterizations and alignment objectives (implementation details in Appendix D). (i) **Align-seqVAE** is based on the SeqVAE alignment framework of Vermani et al. (2024). This framework is most closely related to DiDA, but learns an unconstrained transformation for each new session. Because its original neural-data implementation incorporates behavioral supervision, for unsupervised adaptation we instead use the Gaussian latent-distribution matching objective proposed in NoMAD (Karpowicz et al., 2025). (ii) **ERDiff** (Wang et al., 2023) aligns sessions by modeling source latent trajectory

Table 1: CO velocity decoding performance (R^2) under different adaptation objectives and numbers of observed trials n_{adapt}. Values are reported as mean $\pm$ std across target sessions.

n_{adapt}	Methods	Supervised	Unsupervised	Preference-guided
10	DiDA	$\mathbf{0.6888 \pm 0.1008}$	$\mathbf{0.5232 \pm 0.1536}$	0.6026 ± 0.1363
	Align-seqVAE	0.6516 ± 0.0819	-0.2357 ± 0.3010	–
	ERDiff	0.5674 ± 0.1031	0.0825 ± 0.1476	–
	CycleGAN	0.5783 ± 0.0829	0.3695 ± 0.0855	–
25	DiDA	$\mathbf{0.7535 \pm 0.0912}$	$\mathbf{0.6362 \pm 0.1128}$	0.6463 ± 0.1166
	Align-seqVAE	0.7260 ± 0.1015	-0.1192 ± 0.3187	–
	ERDiff	0.6340 ± 0.0940	0.1053 ± 0.2190	–
	CycleGAN	0.6393 ± 0.0976	0.3592 ± 0.0937	–
50	DiDA	$\mathbf{0.7941 \pm 0.0530}$	$\mathbf{0.6743 \pm 0.0922}$	0.6770 ± 0.0806
	Align-seqVAE	0.7763 ± 0.0571	-0.0510 ± 0.3706	–
	ERDiff	0.7265 ± 0.0503	-0.0099 ± 0.3868	–
	CycleGAN	0.6944 ± 0.0614	0.3683 ± 0.0945	–
Offline	DiDA	0.8201 ± 0.0476	$\mathbf{0.6950 \pm 0.0842}$	–
	Align-seqVAE	$\mathbf{0.8825 \pm 0.0278}$	0.6864 ± 0.0952	–
	ERDiff	0.8159 ± 0.0436	0.6133 ± 0.1120	–
	CycleGAN	0.8306 ± 0.0342	0.5643 ± 0.1187	–

distributions with a diffusion model. We use its original diffusion-based objective for unsupervised adaptation and add behavioral supervision for the supervised setting. (iii) **CycleGAN** (Ma et al., 2023) learns nonlinear neural transformations through adversarial distribution matching and cycle consistency. We adopt its original alignment objectives for unsupervised adaptation and include the behavioral decoding loss for supervised adaptation.

Evaluation Protocol. For both datasets, we use Day 0 as the source session for pretraining and keep the resulting behavioral decoder fixed during adaptation. Each subsequent session is aligned to the Day 0 model, with the final 20 trials reserved as a held-out test set. We evaluate both offline adaptation using all available pre-test trials and online sequential adaptation where trials are incorporated progressively. Adaptation performance is further evaluated at early, intermediate, and late stages, using the R^2 between predicted and ground-truth cursor velocities on the held-out trials.

Oracle preference construction. For preference-guided adaptation, we construct preference signals using oracle behavioral feedback. Specifically, candidate drift states are sampled and the decoded velocities are integrated into cursor paths. The relative quality of the resulting paths is then used to construct preference pairs (details in Appendix B.5).

4.1 Structured drift modeling enables efficient sequential adaptation

In the conventional offline setting, results in Tables 1 and 2 demonstrate that flexible transformations can effectively capture session-specific changes when sufficient adaptation data are available. Under this setting, high-capacity alignment models recover mappings compatible with the Day 0 decoder, resulting in small performance differences across methods. However, this advantage does not consistently extend to sequential adaptation, where only limited trials are initially available. In this regime, DiDA provides more reliable decoding performance, suggesting that highly flexible transformations are difficult to identify from limited observations.

Limited data pose an additional challenge for unsupervised adaptation, as distributional or dynamical statistics estimated from only a few trials may provide a noisy or ambiguous optimization signal. This uncertainty is further amplified when a high-dimensional session-specific transformation must be estimated from the same limited observations. DiDA mitigates both issues by using pretraining to define a lower-dimensional drift space and a shared prior that provides a structured initialization for unseen sessions. Sequential adaptation therefore only needs to identify a small deviation within

Table 2: RT velocity decoding performance (R^2) under different adaptation objectives and numbers of observed trials n_{adapt}. Values are reported as mean ± std across target sessions.

n_{adapt}	Methods	Supervised	Unsupervised	Preference-guided
10	DiDA	$\mathbf{0.5130 \pm 0.1744}$	$\mathbf{0.3047 \pm 0.3509}$	0.4199 ± 0.1321
	Align-seqVAE	0.0306 ± 1.1617	0.0930 ± 0.3875	–
	ERDiff	0.2644 ± 0.2041	-0.0650 ± 0.1417	–
	CycleGAN	-0.1689 ± 1.7536	0.1511 ± 0.4034	–
25	DiDA	$\mathbf{0.4716 \pm 0.3910}$	$\mathbf{0.4093 \pm 0.2549}$	0.4338 ± 0.1644
	Align-seqVAE	0.2243 ± 0.4876	0.0950 ± 0.5761	–
	ERDiff	0.2838 ± 0.3617	-0.2712 ± 0.4397	–
	CycleGAN	-0.4547 ± 2.7603	0.0594 ± 0.5297	–
50	DiDA	$\mathbf{0.5920 \pm 0.0915}$	$\mathbf{0.4444 \pm 0.1884}$	0.4940 ± 0.1068
	Align-seqVAE	0.2783 ± 0.3940	0.1908 ± 0.3001	–
	ERDiff	0.4043 ± 0.2140	-0.2108 ± 0.2977	–
	CycleGAN	-0.2877 ± 2.2509	0.1333 ± 0.4543	–
Offline	DiDA	0.6080 ± 0.0841	0.4650 ± 0.1435	–
	Align-seqVAE	$\mathbf{0.6841 \pm 0.0709}$	0.4647 ± 0.1119	–
	ERDiff	0.6134 ± 0.0651	0.2179 ± 0.0642	–
	CycleGAN	0.5969 ± 0.0673	$\mathbf{0.4746 \pm 0.0781}$	–

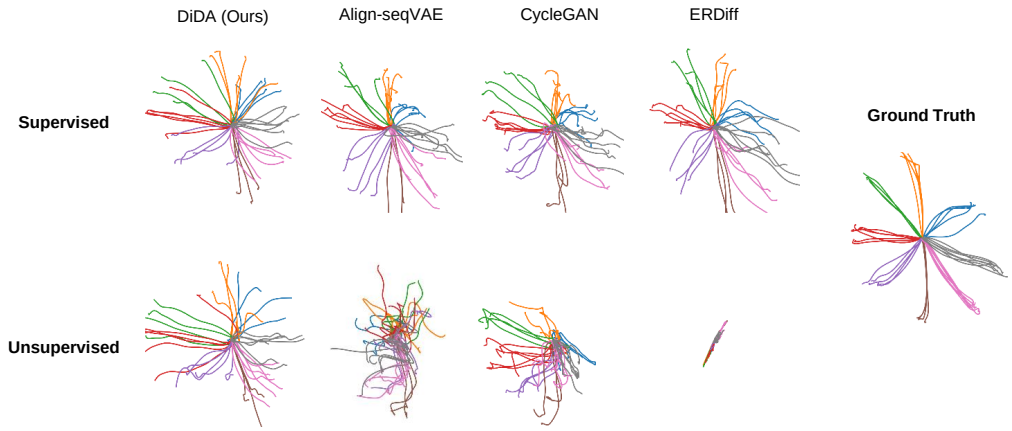

Figure 3: **CO velocity decoding results on the test set**. Results are shown for the final recorded session (38 days after Day 0), after observing 25 trials from this session. Decoded velocities obtained using DiDA, Align-seqVAE, CycleGAN, and ERDiff are compared under supervised (top row) and unsupervised (bottom row) settings. Ground-truth velocities are shown on the left. Different colors represent different movement directions.

this learned space rather than estimate a new transformation from scratch. The contrast between offline and sequential performance thus suggests that effective cross-session adaptation depends not only on model capacity, but also on matching the degrees of freedom being optimized to the amount and reliability of information available during adaptation.

4.2 Different adaptation objectives provide complementary feedback signals

The three adaptation objectives further illustrate how the information available during adaptation affects this process. Supervised adaptation provides the most direct learning signal, since the behavioral loss explicitly indicates how the drift state should change to recover the correct Day 0 decoding mapping. Accordingly, it generally yields the strongest and most stable sequential performance.

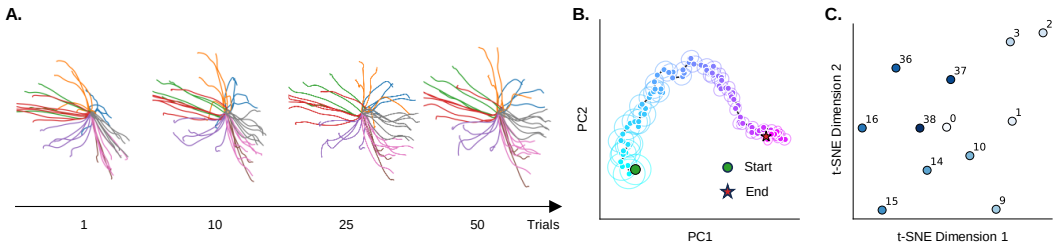

Figure 4: **Preference-guided adaptation through drift-state distribution optimization.** (A) Cursor velocity decoding on test set of the final recorded session (38 days after Day 0) after preference-guided adaptation with increasing numbers of observed trials. (B) Evolution of the drift-state distribution during preference-guided adaptation, visualized by projecting the distributions onto the first two principal components. The trajectory shows the progressive shift of the drift distribution toward preferred regions. (C) t-SNE visualization of the session-specific drift-state distribution means after preference-guided adaptation. The numbers indicate the day offset from Day 0.

Unsupervised adaptation provides a weaker signal: aligning target and source latent distributions constrains population-level statistics but does not directly identify the alignment that improves behavioral decoding. This ambiguity is amplified with few observations, as the behaviorally relevant drift component is difficult to infer. Nevertheless, DiDA maintains robust unsupervised performance, suggesting that restricting adaptation to a learned drift space facilitates drift inference without behavioral labels.

Preference-guided adaptation lies between these two settings in terms of the information supplied to the model. Unlike supervised adaptation, preference feedback does not provide an explicit behavioral target. However, it provides directional information by indicating which of two candidate drift states produces a more desirable behavioral prediction. This signal is more directly related to decoding quality than distributional matching, while remaining weaker than fully supervised behavioral feedback. The consistent improvement of preference-guided adaptation shows that such relative information is able to progressively shift the drift-state distribution toward behaviorally appropriate regions (Figure 4).

5 Discussion and Conclusion

Long-term BCI deployment requires adaptation methods that compensate for neural variability while reducing repeated calibration. Existing cross-session approaches typically learn session-specific alignments, which provide flexible drift correction but often require sufficient target-session data. This motivates us to reframe cross-session recalibration from an offline batch optimization problem into a sequential adaptation problem, where the decoder must recover performance as observations arrive over time. We address this challenge by modeling session variability through a structured probabilistic drift state, enabling efficient online adaptation.

The results show that this formulation provides an effective balance between flexibility and adaptation efficiency. High-capacity alignment models can achieve strong offline performance when sufficient data are available, whereas DiDA is more suitable for sequential adaptation by constraining the degrees of freedom that must be estimated from early-session data. The probabilistic drift representation also enables adaptation under different adaptation signals, including behavioral supervision, unsupervised constraints, and preference-based signals, providing a unified framework for flexible cross-session calibration.

Several limitations suggest directions for future work. While preference-guided adaptation demonstrates the flexibility of the proposed drift representation, the current experiments serve as a proof-of-concept by using oracle behavioral outcomes to construct preference signals. Future studies should investigate how human-in-the-loop feedback can be integrated into closed-loop BCI systems and how noisy or delayed preference signals affect adaptation. Furthermore, the current evaluation focuses on offline recordings, and future work could validate the framework in online closed-loop BCI experiments to better characterize its performance under real-time constraints.

REFERENCES

Iason Batzianoulis, Fumiaki Iwane, Shupeng Wei, Carolina Gaspar Pinto Ramos Correia, Ricardo Chavarriaga, José del R Millán, and Aude Billard. Customizing skills for assistive robotic manipulators, an inverse reinforcement learning approach with error-related potentials. *Communications biology*, 4(1):1406, 2021.

David M Brandman, Tommy Hosman, Jad Saab, Michael C Burkhart, Benjamin E Shanahan, John G Ciancibello, Anish A Sarma, Daniel J Milstein, Carlos E Vargas-Irwin, Brian Franco, et al. Rapid calibration of an intracortical brain–computer interface for people with tetraplegia. *Journal of neural engineering*, 15(2):026007, 2018.

Nicholas S Card, Tyler Singer-Clark, Hamza Peracha, Carrina Iacobacci, Xianda Hou, Maitreyee Wairagkar, Zachery Fogg, Elena C Offenberg, Leigh R Hochberg, Sergey D Stavisky, et al. Long-term independent use of an intracortical brain–computer interface for speech and cursor control. *Nature medicine*, pp. 1–7, 2026.

Ujwal Chaudhary, Niels Birbaumer, and Ander Ramos-Murguialday. Brain–computer interfaces for communication and rehabilitation. *Nature reviews neurology*, 12(9):513–525, 2016.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

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.

Harald Cramér. *Mathematical Methods of Statistics*, volume 9 of *Princeton Mathematical Series*. Princeton University Press, 1946.

Alan D Degenhart, William E Bishop, Emily R Oby, Elizabeth C Tyler-Kabara, Steven M Chase, Aaron P Batista, and Byron M Yu. Stabilization of a brain–computer interface via the alignment of low-dimensional spaces of neural activity. *Nature biomedical engineering*, 4(7):672–685, 2020.

Ali Farshchian, Juan A Gallego, Joseph P Cohen, Yoshua Bengio, Lee E Miller, and Sara A Solla. Adversarial domain adaptation for stable brain-machine interfaces. In *The Seventh International Conference on Learning Representations*, 2019.

R. A. Fisher. On the mathematical foundations of theoretical statistics. *Philosophical Transactions of the Royal Society of London, Series A: Containing Papers of a Mathematical or Physical Character*, 222(594-604):309–368, 01 1922. ISSN 0264-3952. doi: 10.1098/rsta.1922.0009.

Juan A Gallego, Matthew G Perich, Raeed H Chowdhury, Sara A Solla, and Lee E Miller. Long-term stability of cortical population dynamics underlying consistent behavior. *Nature neuroscience*, 23 (2):260–270, 2020.

Stuart Geman, Elie Bienenstock, and René Doursat. Neural networks and the bias/variance dilemma. *Neural computation*, 4(1):1–58, 1992.

Trevor Hastie, Robert Tibshirani, Jerome Friedman, et al. The elements of statistical learning, 2009.

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.

Xuan Ma, Fabio Rizzoglio, Kevin L Bodkin, Eric Perreault, Lee E Miller, and Ann Kennedy. Using adversarial networks to extend brain computer interface decoding accuracy over time. *elife*, 12: e84296, 2023.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35: 27730–27744, 2022.

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, 2018.

Tsam Kiu Pun, Mona Khoshnevis, Tommy Hosman, Guy H Wilson, Anastasia Kapitonava, Foram Kamdar, Jaimie M Henderson, John D Simeral, Carlos E Vargas-Irwin, Matthew T Harrison, et al. Measuring instability in chronic human intracortical neural recordings towards stable, long-term brain-computer interfaces. *Communications Biology*, 7(1):1363, 2024.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

Omid G Sani, Bijan Pesaran, and Maryam M Shanechi. Dissociative and prioritized modeling of behaviorally relevant neural dynamics using recurrent neural networks. *Nature neuroscience*, 27 (10):2033–2045, 2024.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances in neural information processing systems*, 33:3008–3021, 2020.

David Sussillo, Sergey D Stavisky, Jonathan C Kao, Stephen I Ryu, and Krishna V Shenoy. Making brain–machine interfaces robust to future neural variability. *Nature communications*, 7(1):13749, 2016.

Ayesha Vermani, Il Memming Park, and Josue Nassar. Leveraging generative models for unsupervised alignment of neural time series data. In *International Conference on Learning Representations*, volume 2024, pp. 34540–34555, 2024.

Yule Wang, Zijing Wu, Chengrui Li, and Anqi Wu. Extraction and recovery of spatio-temporal structure in latent dynamics alignment with diffusion models. *Advances in Neural Information Processing Systems*, 36:38988–39005, 2023.

A Theoretical Motivation: Adaptation Complexity under Limited Data

A.1 Finite-sample Parameter Estimation and Adaptation Complexity

A central challenge in cross-session BCI adaptation is estimating session-specific changes from a limited number of target-session observations. This problem can be viewed as a finite-sample statistical estimation problem, where the amount of available information must be sufficient to constrain the degrees of freedom of the parameters being adapted.

Let $\theta \in \mathbb{R}^d$ denote the parameters updated during adaptation, where d represents the effective dimension of the adaptation space. Given target-session observations

$$\mathcal{D}_s = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n, \tag{16}$$

the goal is to estimate θ by minimizing an empirical objective:

$$\hat{\theta} = \arg\min_\theta \frac{1}{n} \sum_{i=1}^{n} \ell(f_\theta(\mathbf{x}_i), \mathbf{y}_i), \tag{17}$$

where n denotes the number of target-session observations, f_θ is the adapted model, and $\ell(\cdot,\cdot)$ is the adaptation loss.

Classical statistical estimation theory characterizes the uncertainty of parameter estimation through the Fisher information matrix (Fisher, 1922; Cramér, 1946). For a parametric model with likelihood $p(\mathbf{y}|\mathbf{x},\theta)$, the Fisher information is defined as

$$I(\theta) = \mathbb{E}\left[\nabla_\theta \log p(\mathbf{y}|\mathbf{x},\theta) \nabla_\theta \log p(\mathbf{y}|\mathbf{x},\theta)^\top\right], \tag{18}$$

where $\nabla_\theta \log p(\mathbf{y}|\mathbf{x},\theta)$ represents the sensitivity of the likelihood to changes in the parameters, and the expectation is taken over the data distribution. The Fisher information quantifies how strongly observations constrain different parameter directions. For any unbiased estimator $\hat{\theta}$, the Cramér–Rao lower bound states that

$$\mathrm{Cov}(\hat{\theta}) \succeq I_n(\theta)^{-1}, \tag{19}$$

where $\mathrm{Cov}(\hat{\theta})$ denotes the covariance of the parameter estimator and $I_n(\theta)$ is the Fisher information accumulated from n observations. As the number of observations increases, the accumulated information increases and parameter uncertainty decreases. Conversely, when only a limited number of observations are available, estimating a model with many adaptable degrees of freedom becomes increasingly difficult because the available data provide insufficient constraints on the parameter space.

In cross-session BCI adaptation, this issue arises when directly optimizing a high-dimensional session-specific transformation:

$$\mathbf{x}_{\mathrm{canonical}} = \mathbf{M}_s \mathbf{x}_s, \tag{20}$$

where $\mathbf{M}_s$ contains $O(N^2)$ free parameters for an N-dimensional neural population. Although such a flexible mapping can capture diverse forms of neural drift, estimating all transformation parameters from a small number of target-session trials creates a large adaptation space with many weakly constrained solutions. DiDA instead introduces a structured drift representation:

$$\mathbf{x}_{\mathrm{canonical}} = \mathbf{M}(\mathbf{c}_s)\mathbf{x}_s, \tag{21}$$

where $\mathbf{c}_s$ is a lower-dimensional latent drift variable describing session-specific variation. Therefore, the adaptation problem is transformed from estimating an unconstrained high-dimensional mapping to estimating a compact drift state that modulates a shared transformation. This design follows the statistical principle that the complexity of an adaptive model should be matched to the amount of available calibration data.

A.2 Bias–Variance Trade-off in Adaptation Model Design

The choice of adaptation complexity introduces a fundamental bias–variance trade-off. Consider a target function $f^*(\mathbf{x})$ and an adapted predictor $\hat{f}_{\mathcal{D}}(\mathbf{x})$ learned from a target-session dataset $\mathcal{D}$. The expected prediction error at a given input can be decomposed as

$$\begin{aligned}\mathbb{E}_{\mathcal{D},\epsilon}\left[(y - \hat{f}_{\mathcal{D}}(\mathbf{x}))^2\right] = &\underbrace{\left(\mathbb{E}_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(\mathbf{x})] - f^*(\mathbf{x})\right)^2}_{\text{Bias}^2} \\ &+ \underbrace{\mathbb{E}_{\mathcal{D}}\left[\left(\hat{f}_{\mathcal{D}}(\mathbf{x}) - \mathbb{E}_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(\mathbf{x})]\right)^2\right]}_{\text{Variance}} + \sigma^2,\end{aligned} \tag{22}$$

where $f^*(\mathbf{x})$ denotes the true input–output relationship, $\hat{f}_{\mathcal{D}}(\mathbf{x})$ denotes the predictor learned from a particular calibration dataset $\mathcal{D}$, and σ^2 represents irreducible observation noise (Geman et al., 1992; Hastie et al., 2009).

The bias term reflects the approximation error introduced by restricting the adaptation hypothesis space. When the adaptation model is too constrained, the true session-specific variation may not be represented within the available transformation family, leading to a larger difference between the expected learned predictor and the underlying function:

$$\mathrm{Bias}^2 = \left(\mathbb{E}_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(\mathbf{x})] - f^*(\mathbf{x})\right)^2. \tag{23}$$

Increasing the flexibility of the adaptation model enlarges the set of representable session transformations and can therefore reduce this approximation bias. However, this increased flexibility also introduces a larger set of possible solutions that can explain a finite calibration dataset. When the number of target-session observations is limited, different calibration samples may lead to substantially different adaptation solutions, increasing estimation variance:

$$\text{Variance} = \mathbb{E}_{\mathcal{D}}\left[\left(\hat{f}_{\mathcal{D}}(\mathbf{x}) - \mathbb{E}_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(\mathbf{x})]\right)^2\right]. \tag{24}$$

Therefore, increasing adaptation capacity produces two competing effects: it decreases approximation bias by allowing more expressive session-specific transformations, but increases estimation variance by making the learned adaptation more sensitive to limited calibration samples.

In cross-session BCI adaptation, directly estimating an unrestricted session-specific mapping

$$\mathbf{x}^{(s)}_{\text{adapted}} = \mathbf{M}_s \mathbf{x}^{(s)} \tag{25}$$

provides high flexibility and can capture diverse neural drift patterns. However, the large number of adaptable degrees of freedom makes the estimated transformation highly dependent on the limited observations available in a new session.

DiDA instead restricts session-specific variation to a structured drift space:

$$\mathbf{x}^{(s)}_{\text{adapted}} = \mathbf{M}(\mathbf{c}^{(s)})\mathbf{x}^{(s)}, \tag{26}$$

where $\mathbf{c}^{(s)}$ represents the latent drift state of session s. This restriction may introduce additional assumptions about the form of neural drift, but substantially reduces the variability of the adapted solution under limited data. Importantly, DiDA does not restrict the entire predictive model; shared neural representations, latent dynamics, and behavioral decoding functions remain learned from historical sessions. Therefore, DiDA aims to preserve representation capacity while reducing the variance associated with estimating session-specific changes from sparse target-session observations.

B Implementation Details

B.1 Neural Variational Objective

Latent encoder. The encoder first removes session variability through the context mapping:

$$\widetilde{\mathbf{x}}_t = \mathbf{x}_t M(\mathbf{c}), \tag{27}$$

and then applies a recurrent encoder:

$$h_{1:T} = \text{GRU}(\widetilde{x}_{1:T}). \tag{28}$$

The approximate posterior is parameterized as

$$q_\phi(\mathbf{z}_t|\mathbf{x}_{1:t}, \mathbf{c}) = \mathcal{N}(\mu_z(h_t), \text{diag}(\sigma_z(h_t)^2)). \tag{29}$$

Latent samples are obtained using the reparameterization trick.

Latent dynamics model. The latent dynamics model is parameterized as a Gaussian Markov transition:

$$p_\theta(\mathbf{z}_t|\mathbf{z}_{t-1}) = \mathcal{N}(\mu_\theta(\mathbf{z}_{t-1}), \text{diag}(\sigma^2_\theta)), \tag{30}$$

where the transition mean is implemented using a residual multilayer perceptron:

$$\mu_\theta(\mathbf{z}_{t-1}) = \mathbf{z}_{t-1} + F_\theta(\mathbf{z}_{t-1}). \tag{31}$$

The transition variance is learned as a state-independent diagonal covariance.

Drift-state-conditioned neural observation model. Instead of directly conditioning the observation model on the drift state, we model session variability through a drift-dependent linear transformation similar to the encoder. Given a sampled drift state $\mathbf{c}$, a transformation matrix is generated by a linear mapping:

$$\Delta M(\mathbf{c}) = \mathrm{reshape}(W_c\mathbf{c} + b_c), \tag{32}$$

and the session-specific transformation is defined as

$$M(\mathbf{c}) = I + \Delta M(\mathbf{c}). \tag{33}$$

The nominal neural observation is first generated from the latent state,

$$\mathbf{x}_{\mathrm{nominal}} = f_\xi(\mathbf{z}), \tag{34}$$

and transformed into the session-specific observation space:

$$\boldsymbol{\mu}_x = \mathbf{x}_{\mathrm{nominal}} M(\mathbf{c}). \tag{35}$$

The observation likelihood is modeled as a Gaussian distribution:

$$p_\xi(\mathbf{x}_t|\mathbf{z}_t, \mathbf{c}) = \mathcal{N}(\boldsymbol{\mu}_{x,t}, \mathrm{diag}(\boldsymbol{\sigma}_x^2)), \tag{36}$$

where the observation variance is learned as a global diagonal parameter.

Behavioral decoder. The behavioral decoder maps latent states to movement variables using a linear Gaussian model:

$$p_\psi(\mathbf{y}_t|\mathbf{z}_t) = \mathcal{N}(W_y\mathbf{z}_t + b_y, \mathrm{diag}(\sigma_y^2)). \tag{37}$$

The decoder is trained jointly during pretraining and fixed during cross-session adaptation. The behavioral objective is implemented as the negative log-likelihood:

$$\mathcal{L}_{\mathrm{beh}} = -\mathbb{E}_s \sum_t \log p_\psi(\mathbf{y}_t^{(s)}|\mathbf{z}_t^{(s)}). \tag{38}$$

B.2 Stochastic Approximation of the K-Step Dynamics Term

The K-step predictive density is obtained by marginalizing over the intermediate latent states:

$$p_\theta(\mathbf{z}_t \mid \mathbf{z}_{t-K}) = \int p_\theta(\mathbf{z}_t \mid \mathbf{z}_{t-1}) \prod_{j=1}^{K-1} p_\theta(\mathbf{z}_{t-K+j} \mid \mathbf{z}_{t-K+j-1})\, d\mathbf{z}_{t-K+1:t-1}. \tag{39}$$

Let r_θ denote the rollout distribution over intermediate latent states:

$$r_\theta\left(\widetilde{\mathbf{z}}_{t-K+1:t-1} \mid \mathbf{z}_{t-K}\right) = \prod_{j=1}^{K-1} p_\theta\left(\widetilde{\mathbf{z}}_{t-K+j} \mid \widetilde{\mathbf{z}}_{t-K+j-1}\right). \tag{40}$$

Then

$$p_\theta(\mathbf{z}_t \mid \mathbf{z}_{t-K}) = \mathbb{E}_{r_\theta}\left[p_\theta(\mathbf{z}_t \mid \widetilde{\mathbf{z}}_{t-1})\right]. \tag{41}$$

Applying Jensen's inequality gives

$$\log p_\theta^{(K)}(\mathbf{z}_t \mid \mathbf{z}_{t-K}) = \log \mathbb{E}_{r_\theta}\left[p_\theta(\mathbf{z}_t \mid \widetilde{\mathbf{z}}_{t-1})\right] \geq \mathbb{E}_{r_\theta}\left[\log p_\theta(\mathbf{z}_t \mid \widetilde{\mathbf{z}}_{t-1})\right]. \tag{42}$$

Consequently, stochastic rollouts provide a Monte Carlo estimator of the expected endpoint log-density in equation 42, yielding a tractable lower-bound objective for multi-step temporal regularization.

B.3 Unsupervised Adaptation Objective

For two d-dimensional Gaussian distributions, the KL divergence used for unsupervised adaptation has the closed-form expression

$$\mathcal{L}_{\mathrm{unsup}} = \frac{1}{2}\left[\mathrm{tr}\left(\boldsymbol{\Sigma}_{\mathrm{new}}^{-1}\boldsymbol{\Sigma}_0\right) + \left(\boldsymbol{\mu}_{\mathrm{new}} - \boldsymbol{\mu}_0\right)^\top \boldsymbol{\Sigma}_{\mathrm{new}}^{-1}\left(\boldsymbol{\mu}_{\mathrm{new}} - \boldsymbol{\mu}_0\right) - d + \log\frac{\det\boldsymbol{\Sigma}_{\mathrm{new}}}{\det\boldsymbol{\Sigma}_0}\right]. \tag{43}$$

B.4 DPO Objective for Drift-State Adaptation

For a new session, the drift state is modeled as a diagonal Gaussian distribution,

$$q_\eta(\mathbf{c}) = \mathcal{N}\left(\mathbf{c}; \boldsymbol{\mu}, \mathrm{diag}(\boldsymbol{\sigma}^2)\right). \tag{44}$$

Before DPO adaptation, we retain a fixed copy of the initial drift-state distribution as the reference distribution,

$$q_{\mathrm{ref}}(\mathbf{c}) = \mathcal{N}\left(\mathbf{c}; \boldsymbol{\mu}_{\mathrm{ref}}, \mathrm{diag}(\boldsymbol{\sigma}^2_{\mathrm{ref}})\right). \tag{45}$$

Here, $\boldsymbol{\mu}_{\mathrm{ref}}$ and $\boldsymbol{\sigma}^2_{\mathrm{ref}}$ are the drift-state parameters before adaptation and remain fixed throughout DPO optimization, whereas $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}^2$ are updated during adaptation.

For a candidate drift state $\mathbf{c}$, its log probability under the current distribution is

$$\log q_\eta(\mathbf{c}) = -\frac{1}{2}\sum_{j=1}^{d_c}\left[\log(2\pi) + \log\sigma_j^2 + \frac{(c_j-\mu_j)^2}{\sigma_j^2}\right]. \tag{46}$$

For each preference pair $(\mathbf{c}_i^+, \mathbf{c}_i^-)$, we define

$$\Delta_\eta^{(i)} = \log q_\eta(\mathbf{c}_i^+) - \log q_\eta(\mathbf{c}_i^-), \tag{47}$$

and analogously for the fixed reference distribution,

$$\Delta_{\mathrm{ref}}^{(i)} = \log q_{\mathrm{ref}}(\mathbf{c}_i^+) - \log q_{\mathrm{ref}}(\mathbf{c}_i^-). \tag{48}$$

The DPO objective can then be expressed as

$$\mathcal{L}_{\mathrm{DPO}} = -\frac{1}{N_p}\sum_{i=1}^{N_p}\log\sigma\left(\beta\left[\Delta_\eta^{(i)} - \Delta_{\mathrm{ref}}^{(i)}\right]\right). \tag{49}$$

The reference distribution therefore provides a fixed baseline for measuring how preference optimization changes the relative likelihood of preferred and dispreferred drift states.

B.5 Oracle Preference Construction

At each preference-guided adaptation step, we sample a set of candidate drift states from the current session-specific distribution $q_\eta(\mathbf{c})$. Each candidate drift state is passed through the fixed neural decoder to obtain the corresponding predicted behavioral trajectory. Since the model predicts cursor velocity, we integrate the velocity sequence over time to recover the corresponding cursor trajectory:

$$\hat{\mathbf{p}}_t = \sum_{\tau=1}^{t}\hat{\mathbf{y}}_\tau\Delta t, \tag{50}$$

where $\hat{\mathbf{y}}_\tau$ denotes the predicted velocity and Δt is the sampling interval. The quality of each candidate drift state is evaluated using a trajectory-based oracle score. Specifically, we compute the distance between the predicted and ground-truth cursor trajectories and use its negative value as the preference score:

$$S(\mathbf{c}) = -\frac{1}{T}\sum_{t=1}^{T}\left\|\hat{\mathbf{p}}_t - \mathbf{p}_t\right\|_2. \tag{51}$$

Candidate drift states are ranked according to their scores, and preference pairs are constructed by pairing higher-scoring candidates with lower-scoring candidates. These pairs are then used in the DPO objective to update the session-specific drift distribution.

B.6 Day-0 Signal Augmentation

To enable drift-aware pretraining, we augment the Day-0 neural recordings with synthetic session variations. Specifically, we generate four perturbed copies of the Day-0 neural signals and assign them distinct session identities. The behavioral trajectories and task conditions are kept unchanged, while only the neural observations are modified. These synthetic sessions are designed to mimic

common sources of cross-session neural variability, including changes in signal statistics, channel reliability, electrode correspondence, and spatial recording drift.

For each augmented session, we first apply channel-wise gain and bias perturbations. The gain factor for each channel is sampled as

$$g_i = 1 + \epsilon_i, \quad \epsilon_i \sim \mathcal{N}(0, \sigma_g^2), \tag{52}$$

and the additive bias is generated according to the original channel variance:

$$b_i = \sigma_i \epsilon_i', \quad \epsilon_i' \sim \mathcal{N}(0, \sigma_b^2), \tag{53}$$

where σ_i denotes the standard deviation of channel i in the original Day-0 recordings. The perturbed neural activity is obtained as

$$x'_{t,i} = g_i x_{t,i} + b_i. \tag{54}$$

Based on this statistical perturbation, we further construct three types of session drift. First, **channel dropout drift** randomly removes a subset of neural channels by setting their activities to zero, simulating unreliable or missing neural recordings. Second, **channel permutation drift** randomly swaps pairs of channels, mimicking changes in channel correspondence across recording sessions. Third, **spatial coordinate drift** simulates electrode displacement by shifting electrode locations and applying spatial mixing. Given the original electrode coordinates $\{p_i\}$, we first introduce a spatial displacement:

$$p_i' = p_i + \Delta p, \tag{55}$$

where Δp is a predefined coordinate shift. We then construct a spatial mixing matrix:

$$M_{ij} = \frac{\exp\left(-\frac{\|p_i' - p_j\|^2}{2\sigma_c^2}\right)}{\sum_k \exp\left(-\frac{\|p_i' - p_k\|^2}{2\sigma_c^2}\right)}, \tag{56}$$

and apply it to the neural activity:

$$x'_{t,j} = \sum_i x_{t,i} M_{ij}. \tag{57}$$

After augmentation, all perturbed signals are clipped to non-negative values and combined with the original recordings. The resulting augmented sessions preserve the underlying behavioral structure while introducing diverse session-specific neural variations. These synthetic drifts encourage DiDA to disentangle shared latent dynamics from session-dependent variability during pretraining.

C Dataset

C.1 Dataset description

We used the center-out reaching (CO) and random-target reaching (RT) datasets (Ma et al., 2023). In these datasets, a non-human primate performed reaching tasks while neural population activity and two-dimensional cursor kinematics were simultaneously recorded across multiple sessions using an intracortical microelectrode array. Behavioral measurements were derived from the task cursor kinematics and synchronized to the neural stream. In CO, the monkey reaches from a central position to one of eight predefined targets, yielding stereotyped and directionally balanced trajectories, whereas RT consists of sequential reaches to randomly selected target locations, producing more variable and less predictable movement patterns.

C.2 Data preprocessing

Neural preprocessing. For each trial, neural and behavioral signals were extracted from the task cue to the trial end. Neural spike counts were aggregated by summation within non-overlapping 20-ms bins, while cursor position, velocity, and acceleration were averaged within the same bins. Specifically, for a bin $\mathcal{B}_t$, the binned neural activity and behavioral variables were computed as

$$x_{c,t} = \sum_{\tau \in \mathcal{B}_t} s_{c,\tau}, \qquad y_t = \frac{1}{|\mathcal{B}_t|} \sum_{\tau \in \mathcal{B}_t} y_\tau,$$

where $s_{c,\tau}$ denotes the spike count of channel c at the original time point τ, and y_τ denotes the corresponding behavioral variable. The binned spike counts were subsequently smoothed using a Gaussian kernel with a standard deviation of 40 ms. Trials were truncated to a common duration within each dataset before being used for model training.

Behavioral preprocessing. We inspected cursor trajectories to identify implausible behavioral trials and removed detected outliers. Cursor velocities were normalized using robust z-score normalization based on the median and median absolute deviation (MAD). The MAD was scaled to provide a standard-deviation-equivalent measure under the Gaussian assumption.

D BASELINE IMPLEMENTATIONS

All baseline methods were evaluated under both supervised and unsupervised adaptation settings. For each target session, adaptation was performed using the available training trials, and decoding performance was subsequently evaluated on a held-out test set consisting of the final 20 trials of the session.

D.1 ALIGN-SEQVAE

Align-seqVAE is based on the sequential VAE alignment framework of Vermani et al. (2024). The model learns a session-specific neural transformation to align target-session activity with a shared latent representation. For supervised adaptation, we follow the original implementation and optimize the alignment objective with behavioral supervision. For unsupervised adaptation, since the original neural-data implementation relies on behavioral labels, we replace the behavioral objective with the Gaussian latent-distribution matching objective used in NoMAD (Karpowicz et al., 2025). As Align-seqVAE provides an unconstrained alignment baseline without DiDA's structured drift representation, we keep all shared experimental settings identical to DiDA, except for components specific to the DiDA framework.

D.2 ERDIFF

ERDiff (Wang et al., 2023) combines a recurrent variational autoencoder with a latent diffusion model to learn a compact representation of neural activity. The VAE jointly reconstructs neural signals and predicts cursor velocity, while the diffusion model regularizes the latent dynamics. During target-session adaptation, the pretrained VAE and diffusion model are frozen, and only a session-specific adapter is optimized using latent distribution matching, diffusion regularization. The behavior loss is additionally applied for the supervised mode.

D.3 CYCLEGAN

CycleGAN (Ma et al., 2023) learns bidirectional mappings between the neural activity distributions of a reference session and a target session using adversarial, cycle-consistency, and identity losses. A fixed Wiener decoder trained on the reference session is then applied to the aligned neural activity. For fair comparison, the Wiener decoder uses a single-bin window. The behavior loss is additionally applied for the supervised mode.

E ABLATION STUDY

We perform ablation studies (Appendix Table 1 - 4) to investigate the contribution of different regularization components in DiDA, including the drift-state regularization ($\mathcal{L}_{\mathrm{drift}}$), latent session-invariance regularization ($\mathcal{L}_{\mathrm{adv}}$), and the K-step latent dynamics prior.

In the offline supervised setting, removing individual components results in relatively small performance variations. This is expected because behavioral supervision provides a direct optimization signal for recovering the session-specific mapping, partially compensating for the absence of representation-level constraints. These regularization terms are instead designed to improve the identifiability and stability of the latent drift representation rather than directly optimize behavioral decoding accuracy.

In contrast, the impact of these components becomes more evident under unsupervised and sequential adaptation settings. Removing $\mathcal{L}_{\text{drift}}$ consistently degrades performance, particularly for unsupervised adaptation, indicating that constraining drift distributions toward a shared prior provides a more reliable initialization when behavioral supervision is unavailable. Similarly, removing $\mathcal{L}_{\text{adv}}$ leads to substantial degradation in unsupervised adaptation, suggesting that encouraging session-invariant latent representations helps separate shared behavioral dynamics from session-specific variability.

The K-step dynamics prior also improves robustness during adaptation without explicit behavioral supervision. Although its contribution is less apparent in supervised offline decoding, removing this constraint consistently reduces unsupervised performance across both datasets, with the largest degradation observed in sequential RT adaptation. This suggests that temporal consistency in the latent space provides an additional structural constraint when the available adaptation signal is limited.

Table 1: Ablation study of DiDA components on offline CO velocity decoding. Values denote R^2 scores (mean ± std) evaluated on held-out trials across target sessions.

Methods	Supervised	Unsupervised
DiDA	0.8201 ± 0.0476	$\mathbf{0.6950 \pm 0.0842}$
w/o $\mathcal{L}_{\text{drift}}$	0.8194 ± 0.0468	0.6771 ± 0.0817
w/o $\mathcal{L}_{\text{adv}}$	0.8242 ± 0.0514	0.6801 ± 0.0894
w/o K-step prior	$\mathbf{0.8285 \pm 0.0400}$	0.6815 ± 0.0758

Table 2: Ablation study of DiDA components on offline RT velocity decoding. Values denote R^2 scores (mean ± std) evaluated on held-out trials across target sessions.

Methods	Supervised	Unsupervised
DiDA	0.6080 ± 0.0841	$\mathbf{0.4650 \pm 0.1435}$
w/o $\mathcal{L}_{\text{drift}}$	0.6023 ± 0.0817	0.3947 ± 0.2068
w/o $\mathcal{L}_{\text{adv}}$	$\mathbf{0.6136 \pm 0.0769}$	0.4330 ± 0.1306
w/o K-step prior	0.6070 ± 0.0836	0.3483 ± 0.2947

Table 3: Ablation study of DiDA components on CO velocity decoding performance (R^2) in the sequential adaptation setting with $n_{\text{adapt}} = 25$. Values are reported as mean $\pm$ std across target sessions.

Methods	Supervised	Unsupervised
DiDA	0.7535 ± 0.0912	$\mathbf{0.6362 \pm 0.1128}$
w/o $\mathcal{L}_{\text{drift}}$	$\mathbf{0.7767 \pm 0.0624}$	0.5485 ± 0.3056
w/o $\mathcal{L}_{\text{adv}}$	0.7630 ± 0.0847	0.6352 ± 0.0907
w/o K-step prior	0.7763 ± 0.0667	0.5972 ± 0.1885

F Hyperparameter Analysis

F.1 Update Interval

We further study the effect of adaptation update interval on sequential calibration. While keeping the total adaptation budget ($n_{\text{adapt}} = 25$) and the total number of optimization updates fixed, we vary the interval between consecutive drift-state updates from 1 to 25 trials (Appendix Table 5 - 9).

Across both supervised and unsupervised settings, DiDA remains relatively stable under different update frequencies, achieving strong performance even when updates are performed after a single trial. In contrast, baseline methods generally benefit from using larger update intervals, suggesting

Table 4: Ablation study of DiDA components on RT velocity decoding performance (R^2) in the sequential adaptation setting with $n_{\text{adapt}} = 25$. Values are reported as mean $\pm$ std across target sessions.

Methods	Supervised	Unsupervised
DiDA	0.4716 ± 0.3910	$\mathbf{0.4093 \pm 0.2549}$
w/o $\mathcal{L}_{\text{drift}}$	0.5224 ± 0.1770	0.2864 ± 0.3506
w/o $\mathcal{L}_{\text{adv}}$	$\mathbf{0.5288 \pm 0.2195}$	0.2453 ± 0.6191
w/o K-step prior	0.5258 ± 0.1943	0.2656 ± 0.3655

that their higher-dimensional session-specific transformations require more observations to obtain reliable gradient estimates. Although increasing the update batch size improves baseline performance, their performance remains below DiDA in most settings, especially in the low-data regime with frequent updates.

These results highlight the advantage of structured drift modeling for streaming adaptation: by restricting adaptation to a compact drift distribution, DiDA can extract useful adaptation signals from individual trials, whereas more flexible alignment methods require larger batches to compensate for the increased estimation uncertainty. Therefore, DiDA provides greater flexibility in online calibration scenarios where new observations arrive sequentially.

We also investigate the effect of update interval in preference-guided adaptation (Appendix Table 9). Unlike supervised adaptation, preference-guided adaptation relies on relative preference signals to refine the drift distribution, making timely updates important for incorporating newly available information. The performance consistently decreases as the update interval increases, with the best results achieved when updating after every trial (interval=1) on both CO and RT datasets. This suggests that frequent drift-state updates allow preference signals to be incorporated more effectively, enabling the model to continuously adjust the latent drift distribution during adaptation.

Table 5: CO velocity decoding performance (R^2) under behavioral supervision in the sequential adaptation setting with different update intervals ($n_{\text{adapt}} = 25$). Values are reported as mean $\pm$ std across target sessions.

Update Interval	1	5	10
DiDA	$\mathbf{0.7535 \pm 0.0912}$	$\mathbf{0.7668 \pm 0.0756}$	$\mathbf{0.7646 \pm 0.0697}$
Align-seqVAE	0.7260 ± 0.1015	0.7555 ± 0.0893	0.7329 ± 0.0821
ERDiff	0.6340 ± 0.0940	0.6235 ± 0.1320	0.5927 ± 0.1661
CycleGAN	0.6393 ± 0.0976	0.6418 ± 0.0801	0.6226 ± 0.1050
Update Interval	**15**	**20**	**25**
DiDA	$\mathbf{0.7448 \pm 0.0786}$	$\mathbf{0.7601 \pm 0.0724}$	$\mathbf{0.7760 \pm 0.0688}$
Align-seqVAE	0.7204 ± 0.0727	0.7281 ± 0.0817	0.7380 ± 0.0812
ERDiff	0.5875 ± 0.1424	0.5846 ± 0.1775	0.6180 ± 0.1485
CycleGAN	0.6191 ± 0.0920	0.6203 ± 0.1155	0.6488 ± 0.0968

F.2 Prediction Horizon

We further investigate the effect of the prediction horizon K in the K-step dynamics objective on offline decoding performance (Appendix Table 10 - 11). The prediction horizon controls the temporal range of the latent dynamics regularization, encouraging the learned representation to preserve behaviorally relevant dynamics over different time scales. We focus this analysis on offline decoding because K determines the strength of the latent temporal constraint rather than the adaptation frequency or drift-state parameterization. The offline setting allows us to examine this representation-level effect under a fixed and sufficient adaptation budget.

Across both CO and RT tasks, supervised adaptation remains relatively stable across prediction horizons. In contrast, the effect of K is more pronounced under unsupervised adaptation: CO

Table 6: CO velocity decoding performance (R^2) under the unsupervised mode in the sequential adaptation setting with different update intervals ($n_{adapt} = 25$). Values are reported as mean ± std across target sessions.

Update Interval	1	5	10
DiDA	$\mathbf{0.6362 \pm 0.1128}$	$\mathbf{0.6431 \pm 0.1036}$	$\mathbf{0.6598 \pm 0.1039}$
Align-seqVAE	-0.1192 ± 0.3187	0.3339 ± 0.4021	0.5479 ± 0.1280
ERDiff	0.1053 ± 0.2190	0.2682 ± 0.3207	0.3373 ± 0.1547
CycleGAN	0.3592 ± 0.0937	0.3555 ± 0.0852	0.3557 ± 0.0849
Update Interval	15	20	25
DiDA	$\mathbf{0.6231 \pm 0.1748}$	$\mathbf{0.6620 \pm 0.1093}$	$\mathbf{0.6746 \pm 0.0995}$
Align-seqVAE	0.5788 ± 0.1196	0.6310 ± 0.1068	0.6743 ± 0.0811
ERDiff	0.2870 ± 0.1058	0.3006 ± 0.1434	0.3062 ± 0.1828
CycleGAN	0.3558 ± 0.0849	0.3558 ± 0.0847	0.3554 ± 0.0852

Table 7: RT velocity decoding performance (R^2) under behavioral supervision in the sequential adaptation setting with different update intervals ($n_{adapt} = 25$). Values are reported as mean ± std across target sessions.

Update Interval	1	5	10
DiDA	$\mathbf{0.4716 \pm 0.3910}$	$\mathbf{0.4927 \pm 0.3309}$	$\mathbf{0.4473 \pm 0.4690}$
Align-seqVAE	0.2243 ± 0.4876	0.2621 ± 0.4513	0.2289 ± 0.5007
ERDiff	0.2838 ± 0.3617	0.3012 ± 0.2924	0.1967 ± 0.5004
CycleGAN	-0.4547 ± 2.7603	0.2901 ± 0.3464	-0.0988 ± 1.5245
Update Interval	15	20	25
DiDA	$\mathbf{0.5552 \pm 0.1260}$	$\mathbf{0.3747 \pm 0.6878}$	$\mathbf{0.5394 \pm 0.1878}$
Align-seqVAE	-0.1295 ± 1.1716	0.1946 ± 0.4937	0.2415 ± 0.3696
ERDiff	0.2087 ± 0.4772	0.2184 ± 0.4318	0.3288 ± 0.1880
CycleGAN	0.2319 ± 0.5253	-0.0494 ± 1.3180	0.2312 ± 0.5419

achieves its best performance at $K = 35$, whereas RT peaks at $K = 5$ and degrades substantially for longer horizons. This difference may reflect the temporal structure of the two tasks. CO uses eight predefined target directions and therefore provides relatively structured trajectories, whereas randomly located targets in RT may lead to greater variation in future trajectories. Thus, longer horizons can provide useful temporal constraints when future dynamics are more predictable, but may become less reliable when long-range predictions are more uncertain.

F.3 Drift State Dimension

We investigate the effect of drift-state dimensionality by varying the drift state $\mathbf{c}$ dimension (Appendix Table 12 - 15). For offline supervised adaptation, increasing the drift-state dimension consistently improves performance, as sufficient adaptation data and behavioral supervision allow the model to exploit the additional capacity to capture more complex session-specific variations. In contrast, the benefit of increasing dimensionality is less pronounced for offline unsupervised adaptation. While larger drift spaces improve performance compared with very compact representations, the improvement saturates at higher dimensions, suggesting that latent distribution matching alone provides limited information for identifying more flexible drift states.

For sequential adaptation, the effect of dimensionality depends on the available adaptation signal. Under supervised adaptation, increasing the drift dimension generally improves performance, since behavioral feedback provides a direct constraint on the expanded adaptation space. However, for unsupervised adaptation, excessively large drift spaces do not consistently improve performance, as the drift distribution must be inferred from limited observations without direct behavioral guidance. This highlights the importance of constraining adaptation complexity in low-data regimes.

Table 8: RT velocity decoding performance (R^2) under the unsupervised mode in the sequential adaptation setting with different update intervals ($n_{adapt} = 25$). Values are reported as mean $\pm$ std across target sessions.

Update Interval	1	5	10
DiDA	$\mathbf{0.4093 \pm 0.2549}$	$\mathbf{0.4144 \pm 0.2473}$	$\mathbf{0.4042 \pm 0.2638}$
Align-seqVAE	0.0950 ± 0.5761	0.3120 ± 0.1938	0.1035 ± 0.6823
ERDiff	-0.2712 ± 0.4397	-0.5936 ± 0.7907	-0.8154 ± 1.3340
CycleGAN	0.0594 ± 0.5297	0.1070 ± 0.4687	0.1412 ± 0.4578
Update Interval	15	20	25
DiDA	$\mathbf{0.3873 \pm 0.2992}$	$\mathbf{0.4080 \pm 0.2542}$	$\mathbf{0.4118 \pm 0.2646}$
Align-seqVAE	0.0601 ± 0.8281	0.1726 ± 0.4953	0.1807 ± 0.5413
ERDiff	-0.6829 ± 1.0007	-0.6787 ± 1.4130	-0.4047 ± 0.4706
CycleGAN	0.2595 ± 0.2565	0.2631 ± 0.2488	0.2630 ± 0.2497

Table 9: Velocity decoding performance (R^2) under the preference-guided adaptation mode with different update intervals ($n_{adapt} = 25$) across datasets. Values are reported as mean $\pm$ std across target sessions.

Update Interval	1	5	10
CO	0.6463 ± 0.1166	0.5711 ± 0.1442	0.4877 ± 0.2072
RT	0.4338 ± 0.1644	0.3963 ± 0.1368	0.3455 ± 0.1868
Update Interval	15	20	25
CO	0.4002 ± 0.2497	0.4976 ± 0.1457	0.4557 ± 0.2156
RT	0.2533 ± 0.2227	0.1845 ± 0.4068	0.3034 ± 0.1800

For preference-guided adaptation, the trend is also different from supervised adaptation. Increasing the drift dimension beyond a moderate size does not consistently improve performance, because preference signals only provide relative comparisons between candidate drift states rather than explicit behavioral targets. A larger candidate space therefore increases the difficulty of identifying preferred regions of the drift distribution.

Overall, these results demonstrate that drift-state dimensionality controls the balance between adaptation flexibility and identifiability. Larger drift spaces are beneficial when strong supervision is available, whereas structured and compact drift representations are more reliable when adaptation relies on indirect signals such as latent alignment or preference feedback.

G Implementation details

All experiments were implemented in PyTorch and conducted on a workstation equipped with NVIDIA RTX4090 GPUs. Details of synthetic session generation from Day-0 neural recordings are summarized in Appendix Table 16. The network architecture and training settings across all benchmarks are summarized in Appendix Table 17.

H Statement on the Use of LLM

LLMs were used as supporting tools during the preparation of this work. Specifically, they were used to assist with code development and debugging, as well as language refinement and proofreading of the manuscript. All AI-generated code, suggestions, and textual revisions were critically reviewed, tested, and, where necessary, modified by the author before being incorporated into the final work. AI tools were not used as authoritative sources of information or as substitutes for the author's own analysis and judgment. All substantive research outputs and results were independently verified by the author, who takes full responsibility for the content of the final manuscript.

Table 10: Offline CO velocity decoding performance (R^2) with different prediction horizon K. Values are reported as mean $\pm$ std across target sessions.

K	1	5	15	25	35
Supervised	0.8136 ± 0.0444	0.7894 ± 0.0504	0.8151 ± 0.0448	0.8273 ± 0.0388	0.8201 ± 0.0476
Unsupervised	0.6058 ± 0.1138	0.5634 ± 0.1241	0.6491 ± 0.1119	0.6340 ± 0.0957	0.6950 ± 0.0842

Table 11: Offline RT velocity decoding performance (R^2) with different prediction horizon K. Values are reported as mean $\pm$ std across target sessions.

K	1	5	15	25	35
Supervised	0.5994 ± 0.0738	0.6080 ± 0.0841	0.6094 ± 0.0837	0.6103 ± 0.0848	0.5959 ± 0.0941
Unsupervised	0.4435 ± 0.1031	0.4650 ± 0.1435	0.4170 ± 0.2249	0.3966 ± 0.1395	0.0676 ± 1.0369

Table 12: Offline CO velocity decoding performance (R^2) with different drift-state dimensions. Values are reported as mean $\pm$ std across target sessions.

Drift-state Dimension	16	32	64
Supervised	0.6814 ± 0.0933	0.7774 ± 0.0541	0.8201 ± 0.0476
Unsupervised	0.4853 ± 0.1955	0.5785 ± 0.1420	0.6950 ± 0.0842
Drift-state Dimension	96	128	256
Supervised	0.8476 ± 0.0361	0.8586 ± 0.0297	0.8688 ± 0.0248
Unsupervised	0.6298 ± 0.1335	0.6728 ± 0.0926	0.6787 ± 0.1066

Table 13: Offline RT velocity decoding performance (R^2) with different drift-state dimensions. Values are reported as mean $\pm$ std across target sessions.

Drift-state Dimension	16	32	64
Supervised	0.4959 ± 0.0994	0.5940 ± 0.0838	0.6080 ± 0.0841
Unsupervised	0.2433 ± 0.3664	0.4122 ± 0.2484	0.4650 ± 0.1435
Drift-state Dimension	96	128	256
Supervised	0.6399 ± 0.0733	0.6486 ± 0.0719	0.6352 ± 0.0722
Unsupervised	0.4628 ± 0.1410	0.4601 ± 0.1344	0.4525 ± 0.1471

Table 14: CO velocity decoding performance (R^2) in the sequential adaptation setting with different drift-state dimensions ($n_{adapt} = 25$). Values are reported as mean $\pm$ std across target sessions.

Drift-state Dimension	16	32	64
Supervised	0.6254 ± 0.1356	0.7083 ± 0.0966	0.7535 ± 0.0912
Unsupervised	0.5383 ± 0.1632	0.5881 ± 0.1325	0.6362 ± 0.1128
Preference-guided	0.5709 ± 0.1375	0.6132 ± 0.1537	0.6463 ± 0.1166
Drift-state Dimension	96	128	256
Supervised	0.7907 ± 0.0579	0.8021 ± 0.0458	0.7704 ± 0.0510
Unsupervised	0.5837 ± 0.1565	0.6264 ± 0.1126	0.6257 ± 0.1338
Preference-guided	0.5782 ± 0.1481	0.5932 ± 0.1656	0.5760 ± 0.1432

Table 15: RT velocity decoding performance (R^2) in the sequential adaptation setting with different drift-state dimensions ($n_{adapt} = 25$). Values are reported as mean $\pm$ std across target sessions.

Drift-state Dimension	16	32	64
Supervised	0.4076 ± 0.2359	0.5502 ± 0.1185	0.4716 ± 0.3910
Unsupervised	0.2253 ± 0.4999	0.2797 ± 0.5881	0.4093 ± 0.2549
Preference-guided	0.2677 ± 0.4135	0.4581 ± 0.1089	0.4338 ± 0.1644
Drift-state Dimension	**96**	**128**	**256**
Supervised	0.5664 ± 0.1924	0.5682 ± 0.1636	0.5838 ± 0.0897
Unsupervised	0.3540 ± 0.3700	0.2623 ± 0.6309	0.2521 ± 0.6167
Preference-guided	0.4561 ± 0.1175	0.4296 ± 0.1233	0.4127 ± 0.1205

Table 16: Details of synthetic session generation from Day-0 neural recordings. All augmented sessions share the same gain and bias perturbations, while additional drift types are introduced separately.

Synthetic Session	Additional Perturbation	Parameter	Value
Session 1	Gain + Bias	σ_g, σ_b	0.05, 0.05
Session 2	Channel dropout	Dropout ratio	0.10
Session 3	Channel permutation	Swap ratio	0.10
Session 4	Spatial coordinate drift	Δp, σ_c	$(0.4, -0.25)$, 0.75

Table 17: Architectural and training settings for DiDA.

Architectural settings		**Training settings**	
Latent state dimension	32	Optimizer	Adam
Drift state dimension	64	Adaptation learning rate	1e-2
Encoder hidden dimension	64	Adaptation steps per trial	25
Dynamics MLP hidden dimension	64	Adaptation update interval	1
Dynamics MLP hidden layer number	2	DPO candidate number	1000
Dynamics MLP activation function	Tanh	Preference pairs	8
Adversarial session classifier hidden dimension	64	Prediction horizon (CO)	35
Adversarial session classifier hidden layer number	1	Prediction horizon (RT)	5
Adversarial session classifier activation function	ReLU	Train/validation/test set ratio for pretrain/offline	$0.6/0.2/0.2$