PLASTICITY-COMPATIBLE REPRESENTATIONS ENABLE FAST ADAPTATION THROUGH WRITABLE INTERFACES

Anonymous authors
Paper under double-blind review

ABSTRACT

Long-lived adaptive systems may retain useful internal computation even as their observation or output mappings change during deployment. A central question is therefore not only how to adapt, but what learned representations make simple, restricted adaptation effective. We study a two-timescale recurrent framework in which outer-loop task learning establishes persistent dynamics and a low-dimensional query representation, while deployment-time feedback updates only a small, resettable interface. We show that the geometry of the learned queries directly governs the functional consequences of local plasticity: a feedback-driven write at one state transfers to another in proportion to their query similarity, determining both useful generalization and collateral interference. This yields a mechanistic criterion for plasticity-compatible representations, which we validate through analytic characterization and targeted interventions on the learned write/read geometry. On held-out action remappings, the resulting reward-modulated interface adapts rapidly from sparse binary feedback and remains robust when feedback is corrupted, without modifying the recurrent backbone. We further test the principle on neural population recordings, where writable interfaces support retrospective adaptation to a contingency reversal and cross-day recording drift by targeting output- and observation-side mismatches, respectively. Together, these results show that learned representations determine not only what can be read out from a fixed computation, but also how subsequent local learning propagates through it, providing a route toward selective adaptation without relearning reusable dynamics. The source code of this study is available at: `https://anonymous.4open.science/r/plasticity_compatible_representation-641D`.

1 INTRODUCTION

Long-lived adaptive systems must often operate under changing deployment conditions. For example, a brain-computer interface (BCI) may need to support the same movement objective across days despite changes in recorded neural activity or decoder calibration (Sussillo et al., 2016; Degenhart et al., 2020; Ma et al., 2023; Karpowicz et al., 2025) (Figure 1). An assistive device or robot may retain a useful controller while its sensor calibration, tool, or actuator response changes (Hu et al., 2025). In such cases, the model's existing computation may remain useful, even though its mapping to new inputs or outputs needs to adapt. Yet full-model fine-tuning requires end-to-end backpropagation during deployment and may unnecessarily modify reusable computation (Ding et al., 2023; Dohare et al., 2024), and obscures which adaptation actually restored behavior. This motivates a modular question: *can a system preserve its learned computation and adapt only the interface affected by the deployment shift?*

Existing fast-adaptation methods primarily focus on how information is updated or stored during deployment. Here, we ask a complementary question: what learned representation makes a simple, fixed local update effective? Recurrent meta-RL stores adaptation in distributed activation histories (Duan et al., 2016; Wang et al., 2016; Laskin et al., 2023; Grigsby et al., 2024; Zisman et al., 2024; Nikulin et al., 2025), while fast-weight (Ba et al., 2016; Sun et al., 2024; Behrouz et al., 2025) and differentiable-plasticity approaches introduce rapidly changing states or learned update mechanisms (Miconi et al., 2018; 2019; Tyulmankov et al., 2022; Giannakakis et al., 2025). A representation sufficient for the source task, however, need not make subsequent local updates selective. We therefore fix the deployment-time update rule and ask whether outer-loop task learning can organize

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.

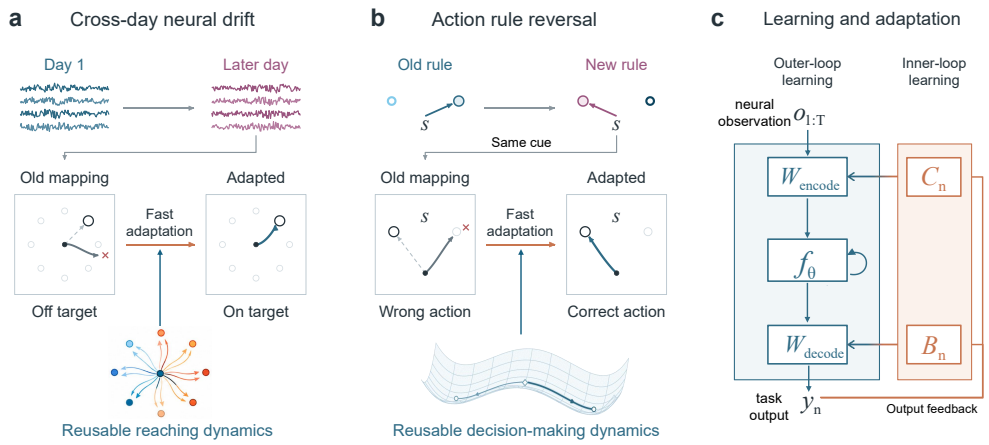

Figure 1: **Outer-loop learning preserves reusable dynamics, whereas inner-loop feedback adapts the task interface. a.** Across-day neural drift can change recorded activity while preserving the underlying reaching goal and reusable dynamics. **b.** Action-rule remapping changes the required action for the same cue, creating a complementary fast-adaptation problem. **c.** The outer loop learns persistent encoding, recurrent dynamics, and slow decoding parameters. During evaluation, these parameters are frozen and feedback updates only a temporary interface: C_n for observation-side alignment or B_n for query-to-action remapping. Thus, adaptation modifies the task interface while preserving the learned recurrent substrate.

a representation on which it becomes effective. Specifically, we use a three-factor rule in which activity determines where to write, an action-related signal determines the update direction, and signed feedback gates the write (Gerstner et al., 2018; Bellec et al., 2020).

We introduce a two-timescale recurrent framework in which outer-loop learning shapes reusable task representations, while deployment-time feedback updates only a small, writable interface (Figure 1). When the required action mapping changes, trial feedback updates only the mapping from learned queries to actions, whereas the recurrent model remains fixed. Whether such restricted updates are useful depends critically on the query geometry: an update should transfer across states that require a shared correction while avoiding interference with states that do not. We call this property *plasticity compatibility*. We also consider shifts that occur on the other side of the recurrent computation. When cross-day neural drift changes the observations entering the model, output remapping alone cannot restore the original representation. We therefore introduce an observation-side alignment state that adapts incoming neural activity to the frozen recurrent dynamics.

Our contributions are summarized as follows:

- **Plasticity-compatible representations.** We show that learned query geometry determines how local updates transfer across states, providing a functional criterion for selective adaptation beyond source-task performance.
- **A mechanistic account of local adaptation.** We derive how query similarity governs the propagation and interference of feedback-driven writes, and validate this mechanism through controlled interventions.
- **Interface-specific adaptation.** We demonstrate rapid output remapping from sparse and noisy feedback, and observation-side recalibration under cross-day neural drift, while preserving the recurrent model.

2 Related Work

Meta-learning and in-context adaptation. Gradient-based meta-learning trains model parameters for rapid adaptation through a small number of gradient updates (Finn et al., 2017; Hospedales et al., 2022). Analyses of MAML highlight the importance of feature reuse: ANIL restricts inner-loop

adaptation to a task-specific head while matching MAML on the evaluated benchmarks (Raghu et al., 2020). Recurrent meta-RL offers another mechanism, learning policies whose hidden-state dynamics implement adaptation from observations, actions, and rewards (Duan et al., 2016; Wang et al., 2016). Transformers can likewise be meta-trained for in-context learning (Kirsch et al., 2022). In reinforcement learning, this capability has been developed through distillation of learning histories (Laskin et al., 2023; Zisman et al., 2024) and end-to-end training of long-context policies (Grigsby et al., 2024). Our framework separates within-trial recurrent processing from across-trial adaptation, storing feedback-driven corrections in an explicit, resettable interface.

Fast weights, parameter-efficient tuning, and test-time memory. Fast weights provide rapidly changing associative memory alongside slowly learned parameters (Ba et al., 2016), and linear attention admits a fast-weight interpretation based on outer-product writes and associative reads (Schlag et al., 2021). Parameter-efficient tuning, including adapters and LoRA, learns a small set of task-specific parameters while keeping the pretrained backbone fixed (Houlsby et al., 2019; Hu et al., 2022; Ding et al., 2023). Test-Time Training (TTT) layers use model parameters as recurrent state, updated through self-supervised learning (Sun et al., 2024), while Titans updates a neural memory module at test time to retain historical context (Behrouz et al., 2025). We investigate restricted interface updates for correcting observation or action mappings under deployment shifts, with the interface location chosen to match the affected mapping.

Learned and neuromodulated plasticity. Three-factor plasticity combines local pre- and post-synaptic activity with an additional modulatory signal, while eligibility traces allow modulation to act on earlier activity (Gerstner et al., 2018). Differentiable Plasticity and Backpropamine optimize plasticity coefficients and neuromodulatory mechanisms through gradient-based outer-loop learning (Miconi et al., 2018; 2019). Related work learns local update rules (Najarro & Risi, 2020; Shervani-Tabar & Rosenbaum, 2023), feedback pathways (Lindsey & Litwin-Kumar, 2020), and memory addressing (Tyulmankov et al., 2022). Task structure and architectural bottlenecks can also shape the reward-modulated rules discovered by meta-learning (Giannakakis et al., 2025). Complementary approaches such as e-prop use eligibility traces to support online learning of recurrent network weights (Bellec et al., 2020). Our work uses prescribed feedback-modulated updates at restricted interfaces of a frozen recurrent model, focusing on the representational and feedback conditions that enable selective adaptation.

Adaptive neural decoding. Stable behavior can coexist with drift in recorded neural activity, while task-relevant latent dynamics remain consistent (Gallego et al., 2020). Motor memories can also be maintained through stable, context-specific population representations (Kim et al., 2025). Approaches to robust neural decoding include training across recording conditions and synthetic perturbations (Sussillo et al., 2016), aligning low-dimensional neural spaces (Degenhart et al., 2020), adversarial distribution alignment (Ma et al., 2023), and alignment to learned latent dynamics (Karpowicz et al., 2025). Closed-loop decoder adaptation can also interact with neural plasticity to support neuroprosthetic control (Orsborn et al., 2014). Complementing unsupervised alignment approaches, we investigate whether limited trial-level correctness feedback can guide interface adaptation around a fixed pretrained decoder backbone. Our neural replay experiments evaluate decoder adaptation on previously recorded activity, rather than closed-loop co-adaptation with the subject.

3 METHOD

3.1 PROBLEM SETTING

On trial n, the model receives a sequence $X_n = (x_{n,1}, \ldots, x_{n,T})$, with $x_{n,\tau} \in \mathbb{R}^D$, and produces a task-specific prediction $\hat{y}_n$. During deployment, the relationship between observations, underlying task states, and desired outputs may change. Here, we consider shifts that can be accommodated by recalibrating a task-specific interface while retaining the pretrained recurrent computation.

After each prediction, the model may receive a feedback signal y_n that provides information about its performance. Feedback is available at the trial level: each complete sequence yields one prediction and at most one feedback event. Let θ denote the persistent model parameters and W_n the writable interface on trial n. During deployment, θ remains fixed, while W_n may be updated using locally

available activity and feedback. The objective is to recover task performance from limited feedback following a deployment shift. We ask what representational properties after outer-loop learning can be established to make this constrained adaptation effective.

3.2 Persistent Recurrent Representation and Outer-loop Learning

The persistent model integrates observations within each trial and maps the resulting recurrent state to a low-dimensional query used by the writable interface. An input projection first maps each observation into the recurrent input space:

$$u_{n,\tau} = P_\theta x_{n,\tau}, \tag{1}$$

where P_θ is learned during outer-loop training or set to the identity when no projection is used. The recurrent state evolves as

$$h_{n,\tau} = \mathrm{GRU}_\theta(\widetilde{u}_{n,\tau}, h_{n,\tau-1}), \qquad h_{n,0} = 0. \tag{2}$$

Here, $\widetilde{u}_{n,\tau} = u_{n,\tau}$ unless an observation-side interface transforms the input, as described below. Resetting $h_{n,0}$ ensures that recurrent activity integrates information within a trial, while the writable interface carries adaptation across trials.

We extract a query from the terminal recurrent state using a normalized linear readout:

$$z_n = \frac{W_q h_{n,T} + b_q}{||W_q h_{n,T} + b_q||_2 + \epsilon}, \qquad z_n \in \mathbb{R}^K, \tag{3}$$

where $\epsilon > 0$ ensures numerical stability. The query provides a shared coordinate for reading and writing task-specific corrections. Temporal processing is performed by the GRU, while the linear query head exposes the representation geometry on which local plasticity acts. Normalization reduces the influence of query magnitude on interface updates.

Further, a persistent readout produces the baseline output:

$$\ell_n^{\mathrm{slow}} = W_s z_n + b_s. \tag{4}$$

The input projection, GRU, query head, and persistent readout are trained in the outer loop and remain fixed during deployment.

Across all three tasks, the outer loop first learns a source-competent persistent decoder before any fast adaptation is evaluated. Given labelled source trials $\mathcal{D}_{\mathrm{src}} = \{(X_n, y_n)\}_{n=1}^N$, the common supervised objective is

$$\mathcal{L}_{\mathrm{outer}}(\Theta) = -\frac{1}{N}\sum_{n=1}^{N} \log \mathrm{softmax}\left(\ell_\Theta^{\mathrm{slow}}(X_n)\right)_{y_n}, \tag{5}$$

where Θ denotes the persistent observation encoder, recurrent model, query head, and slow readout used in that experiment.

3.3 Fast Query-to-Action Interface

For action remapping, we introduce a writable matrix $B_n \in \mathbb{R}^{A\times K}$, where A is the number of actions and K is the query dimension. The interface is initialized to zero at the start of each evaluation session, $B_1 = 0$, and carries feedback-driven updates across trials within that session. It contributes an additive correction to the persistent action logits:

$$\ell_n = \ell_n^{\mathrm{slow}} + gB_n z_n, \quad \pi_n = \mathrm{softmax}(\ell_n), \qquad a_n \sim \pi_n, \tag{6}$$

where $g > 0$ controls the contribution of the fast interface. Each row of B_n defines a linear correction to one action's score as a function of the query. Updating B_n therefore changes the query-to-action mapping while leaving the recurrent dynamics, query projection, and persistent readout fixed.

3.4 Binary Feedback and Local Writing

After each prediction, the environment returns a binary correctness signal:

$$\delta_n = \begin{cases} +1, & a_n = a_n^\star, \\ -1, & a_n \neq a_n^\star, \end{cases} \tag{7}$$

where the correct action $a_n^\star$ is not revealed to the model.

We construct a normalized contrast between the sampled action and the current policy:

$$d_n = \frac{\mathbf{e}_{a_n} - \pi_n}{\max(||\mathbf{e}_{a_n} - \pi_n||_2, \varepsilon)}, \tag{8}$$

where $\mathbf{e}_{a_n}$ is the one-hot action vector. The contrast assigns a positive component to the sampled action and negative components to alternatives, weighted by their policy probabilities.

It follows the softmax policy-score direction $\nabla_{\ell_n} \log \pi_n(a_n) = \mathbf{e}_{a_n} - \pi_n$, with unit norm whenever the numerical floor is inactive. The local eligibility binds this action contrast to the query:

$$E_n = d_n z_n^\top. \tag{9}$$

Signed feedback then modulates the write interface in a three-factor plasticity manner:

$$B_{n+1} = \lambda_B B_n + \eta \delta_n E_n, \tag{10}$$

where $\eta > 0$ is the write rate and $\lambda_B \in [0, 1]$ controls retention. The three factors have distinct roles: z_n determines where to write, the normalized action contrast determines how the action rows change, and δ_n determines the sign. Positive feedback reinforces the sampled action relative to its alternatives, whereas negative feedback reverses the correction. Normalization preserves the direction of the softmax score $\mathbf{e}_{a_n} - \pi_n$ while removing its confidence-dependent magnitude when the numerical floor is inactive. The write interface is therefore set by η and the query norm, while λ_B controls the persistence of previous writes. All persistent parameters remain fixed, and the update requires no backpropagation through the recurrent backbone.

3.5 Query Kernel and Plasticity Compatibility

For one unprojected write at trial i, let $\Delta B_i = \eta v_i z_i^\top$, where v_i contains the signed action correction. At a later trial j,

$$\Delta \ell_j^{\text{fast}} = g \Delta B_i z_j = g \eta v_i \underbrace{(z_i^\top z_j)}_{K_{ij}}. \tag{11}$$

The query Gram matrix K_{ij} is therefore the state-space operator through which a local update generalizes. Feedback determines the action direction v_i, and query geometry determines where the update is expressed. Because queries are normalized, the kernel is their cosine similarity (up to the stabilizing ϵ):

$$K_{ij} = z_i^\top z_j \approx \cos(W_q h_{i,T}, W_q h_{j,T}), \qquad K_{ij} \in [-1, 1]. \tag{12}$$

Positive similarity transfers the same action correction to a later state; orthogonal queries are approximately protected from one another; a query collision broadcasts an indistinguishable correction and therefore cannot support conflicting remappings; and negative similarity reads the stored correction with reversed sign. Negative similarity can be useful for genuinely opponent corrections, but otherwise causes negative transfer.

Normalization also makes collateral interference directly measurable. For a fixed write direction v_i,

$$\mathcal{I}(i, j) = \frac{\|\Delta \ell_j^{\text{fast}}\|_2}{\|\Delta \ell_i^{\text{fast}}\|_2 + \epsilon} \approx |z_i^\top z_j| = |K_{ij}|. \tag{13}$$

The magnitude measures how strongly a local write leaks to another state, while the sign of K_{ij} distinguishes same-direction from reversed transfer. This connects representation geometry to a functional consequence of plasticity rather than treating a low-dimensional projection as evidence by itself.

3.6 Observation-Side Alignment for Neural Drift

The action-side state B_n can change how a query is mapped to an action, but it cannot repair a shift that occurs before the query is formed (e.g., cross-day neural drift). Here, we use a factorized two-stage protocol for cross-day neural drift. The first stage uses unlabeled target observations to estimate a frozen day-specific input coordinate. The second stage uses post-trial feedback to write a small, resettable residual before the frozen recurrent dynamics.

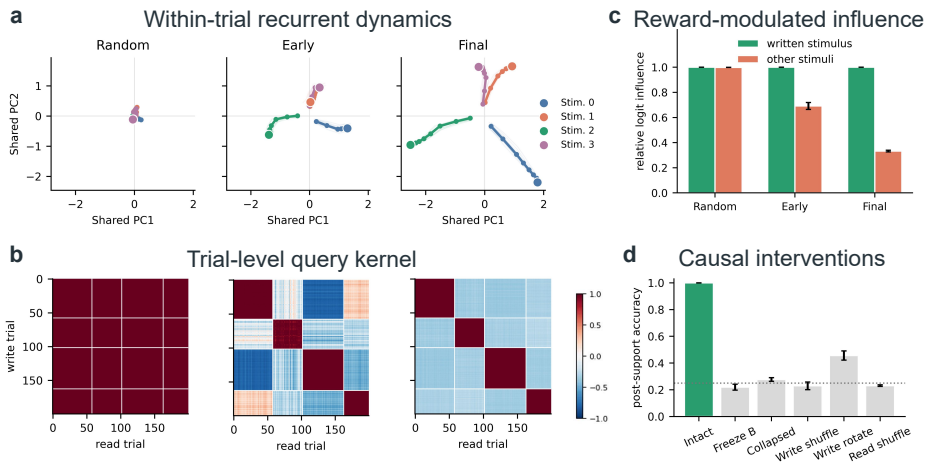

Figure 2: **Outer-loop learning shapes the routing geometry of local plasticity. a.** Within-trial recurrent trajectories at random, early, and final outer-loop checkpoints, projected into a shared PCA space. **b.** The corresponding trial-level query kernels. Outer-loop learning increases within-stimulus consistency and separates queries associated with different actions. **c.** Relative logit influence of one controlled write on the written versus other stimuli. Training preserves the on-target effect while reducing collateral influence. **d.** Post-support remapping accuracy after interventions to the query interface. Freezing B_t, collapsing the queries, or disrupting the write/read coordinate correspondence eliminates the adaptation advantage. The dotted line denotes chance.

For target day d, $X_n = (x_{n,1}, \ldots, x_{n,T})$ is a spike-rate sequence from trial n. Source statistics (μ_s, σ_s) are fitted once on the persistent-training sessions. The first K chronological target trials are used without labels to estimate target channel statistics. With $\tau_C = 10$, and $\alpha_K = K/(K + \tau_C)$, $\widehat{\mu}_{d,K} = (1 - \alpha_K)\mu_s + \alpha_K \mu_{d,K}^{\text{targ}}$, and $\widehat{\sigma}_{d,K} = (1 - \alpha_K)\sigma_s + \alpha_K \sigma_{d,K}^{\text{targ}}$. Here $(\mu_{d,K}^{\text{targ}}, \sigma_{d,K}^{\text{targ}})$ pool all time bins in the K support trials. The aligned observation is

$$\widetilde{x}_{n,\tau} = \frac{x_{n,\tau} - \widehat{\mu}_{d,K}}{\max(\widehat{\sigma}_{d,K}, \varepsilon)}. \tag{14}$$

The statistics state $C_{d,K}^{\text{stat}} = (\widehat{\mu}_{d,K}, \widehat{\sigma}_{d,K})$ is then frozen. It uses no target direction, cursor trajectory, decoder action, day identifier, or correctness signal.

The second stage is a low-dimensional feedback-driven input residual. Let P_θ be the source-trained input projection, and let V be a fixed source-only activity basis. For each time bin, the temporary interface forms

$$p_{n,\tau} = P_\theta \widetilde{x}_{n,\tau} + (\widetilde{x}_{n,\tau} V) A_n^\top + b_n. \tag{15}$$

A_n, b_n are fast adaptation factors at the second stage. The residual starts at $A_0 = 0, b_0 = 0$ and is reset for every target day. Before feedback, the current policy provides an eligibility direction. For binary feedback, let $a_n \sim \pi_n$ and define

$$(C_n^A, C_n^b) = \nabla_{(A_n, b_n)} \log \pi_n(a_n), \quad s_n = \sqrt{\|C_n^A\|_F^2 + \|C_n^b\|_2^2} + \varepsilon, \quad (E_n^A, E_n^b) = s_n^{-1}(C_n^A, C_n^b). \tag{16}$$

The derivatives are computed through the frozen projection, GRU, query head, and readout, but only the temporary residual is changed. The environment then returns $\delta_n = +1$ when $a_n = y_n$ and $\delta_n = -1$ otherwise. The fast update is

$$A_{n+1} = \text{clip}_{[-2,2]}\left(\lambda_A A_n + \eta_A \delta_n E_n^A\right), \quad b_{n+1} = \text{clip}_{[-2,2]}\left(\lambda_A b_n + \eta_A \delta_n E_n^b\right). \tag{17}$$

Direction-resolved feedback replaces the selected action in Eq. 16 by the target direction y_n and uses a positive signal. It is supervised calibration, not binary reward-only feedback. The update is three-factor in the operational sense of eligibility, feedback, and a temporary state, but it is not claimed to be backpropagation-free: autodifferentiation is used only to compute eligibility through a frozen model, there is no persistent gradient update, and no optimizer momentum or other optimizer state is retained. This is a bounded, feedback-gated fast-state adapter rather than full fine-tuning.

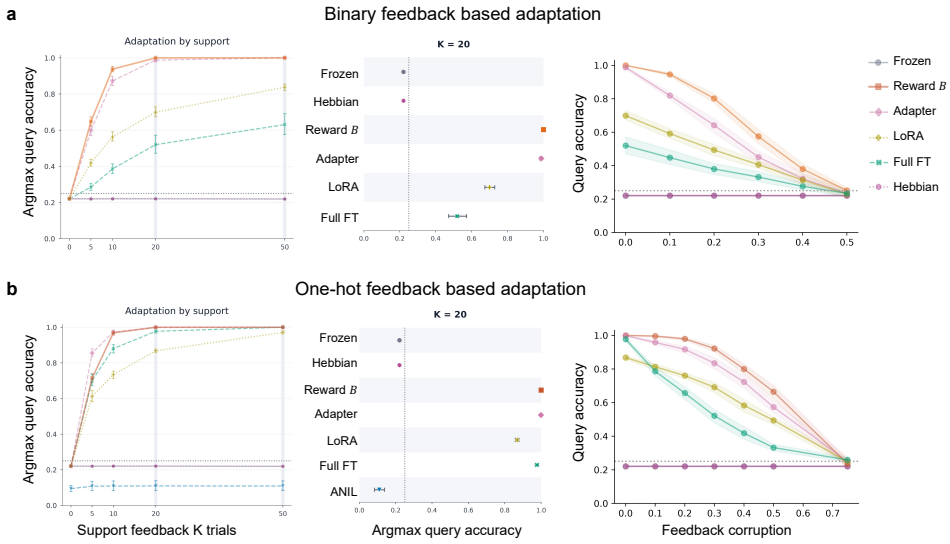

Figure 3: **Fast adaptation from sparse and noisy feedback. a.** Binary-feedback adaptation. Left, query accuracy as a function of the number of chronological support trials. Middle, performance at $K = 20$ across independent seeds. Right, robustness to independently corrupted feedback. **b.** One-hot correct-action feedback under the same three evaluations. Reward-modulated fast connectivity adapts rapidly with few support trials and degrades more slowly under feedback corruption than the matched baseline methods.

4 Experiments

4.1 Outer-loop learning organizes a plasticity-compatible query geometry

We first asked whether outer-loop task learning organizes the query representation in a way that supports selective deployment-time plasticity. Although the outer loop is trained only for source-task performance, the learned queries become increasingly structured over training (Fig. 2a,b). Queries associated with the same stimulus become more consistent, while states associated with different actions become more separated. This organization changes the geometry on which the fixed local update operates.

We next tested whether this geometry predicts the functional consequences of an individual write. As derived above, a write generated at state i affects another state j in proportion to their query similarity, $z_i^\top z_j$. Consistent with this prediction, controlled writes through the learned queries retained a strong effect on the intended state while reducing collateral effects on states requiring different corrections (Fig. 2c). Thus, outer-loop learning did not simply reduce the magnitude of fast updates; it made their effects more selective across states.

Finally, we directly perturbed the write/read geometry to test whether this structure was necessary for fast adaptation. Freezing the fast interface removed adaptation, while collapsing the query representation or disrupting the correspondence between where updates were written and later read out substantially impaired performance (Fig. 2d). Together, these interventions show that the adaptation advantage depends on the learned geometry through which local writes propagate, rather than only on the presence of an additional fast parameter state.

4.2 Fast adaptation from sparse and noisy feedback

We next asked whether this selective write geometry translates into rapid adaptation to a held-out action mapping. During each target session, the recurrent model and persistent readout remained fixed, and only the temporary fast state was updated from chronologically observed feedback. We compared the reward-modulated writer with frozen decoding, local Hebbian variants, and gradient-based adapters under matched support trials and feedback information.

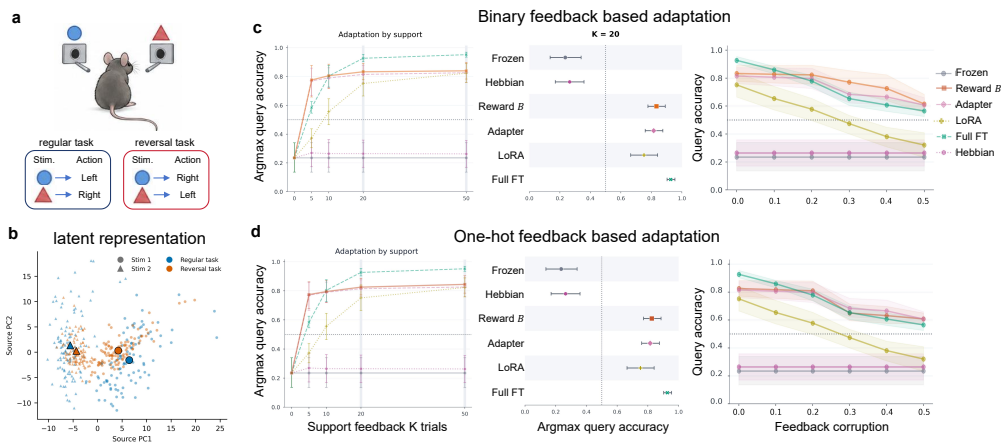

Figure 4: **Fast adaptation to a natural neural contingency reversal.** **a**, Regular and reversed stimulus–action contingencies in the tactile discrimination task. **b**, PCA projection of trial-aligned ALM activity in the two contexts. **c**, Query accuracy after binary correctness feedback. The reward-modulated fast connection (*Reward B*) rapidly improves with a small number of support trials, whereas frozen and Hebbian controls remain near baseline. **d**, The corresponding result with one-hot correct-action feedback.

Under binary correctness feedback, the reward-modulated writer adapted rapidly as feedback accumulated (Fig. 3a,b). Its advantage was already apparent with only a small number of target-session trials, and by $K = 20$ it approached its asymptotic performance. The same pattern held with direction-resolved one-hot feedback, where the writer again reached near-ceiling performance from limited support (Fig. 3 b). In contrast, frozen decoding could not accommodate the new mapping, while the matched local and gradient-based baselines generally required more feedback or achieved lower performance over the same adaptation budget.

We then corrupted the deployment feedback to test whether rapid adaptation depended on perfectly reliable supervision. Performance degraded gradually as feedback noise increased, but the reward-modulated writer retained an advantage over the matched alternatives across the tested corruption levels (Fig. 3). Together, these results show that a small resettable fast state provides rapid and comparatively noise-robust adaptation from sparse feedback, while requiring neither deployment-time backpropagation.

4.3 Adapting to contingency reversal in neural population activity

We next asked whether the output-side writable interface could extend from controlled remapping to a contingency change in real neural population activity. Using a public calcium-imaging dataset, we trained the recurrent decoder on the regular tactile-discrimination contingency and froze its persistent parameters before adapting to the reversed condition. The reversal substantially reduced frozen-decoder performance, as the same sensory evidence now required a different action mapping. With only a small number of binary correctness signals, the reward-modulated interface rapidly recovered performance, whereas frozen decoding and Hebbian adaptation remained near baseline (Fig. 4). Under direction-resolved one-hot feedback, it remained competitive with supervised adapters and full fine-tuning without persistent parameter updates. These retrospective results show that a small output-side state can compensate for a contingency-induced neural-decoder mismatch without modifying the recurrent backbone.

4.4 Observation-side adaptation compensates for cross-day neural drift

The preceding experiments considered changes in the mapping from a reusable recurrent representation to actions. Cross-day neural recording drift presents a complementary case in which the mismatch can arise upstream, between observed neural activity and the frozen recurrent model. We therefore tested whether adaptation at the observation side could recalibrate held-out recording days

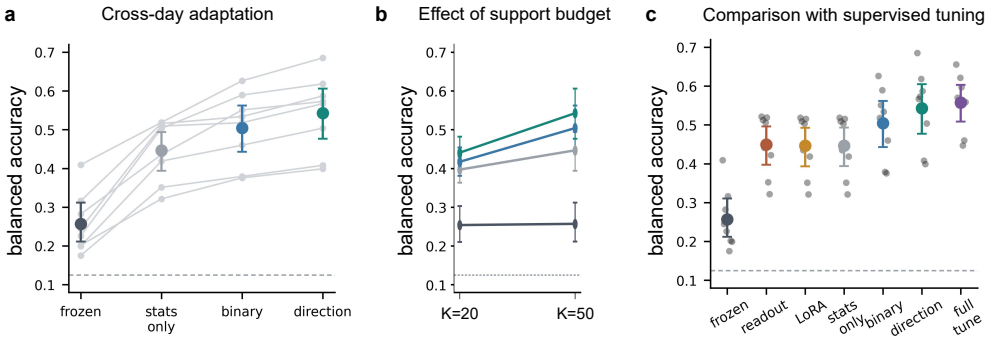

Figure 5: **Cross-day adaptation and supervised comparisons. a.** Balanced accuracy across eight held-out recording days at $K = 50$ for frozen decoding, statistics-only alignment, binary feedback, and direction-resolved feedback. **b.** Effect of the support budget, comparing $K = 20$ and $K = 50$. **c.** Comparison with supervised readout fine-tuning, LoRA, and full fine-tuning at $K = 50$. Points show recording-day means and error bars denote day-level bootstrap 95% confidence intervals.

while preserving the recurrent dynamics. Across eight held-out sessions, statistics-only alignment improved substantially over the frozen model, and binary feedback produced a further gain (0.504 vs. 0.447, $p = 0.0078$; Fig. 5). Direction-resolved feedback further increased performance to 0.543, approaching full fine-tuning (0.558), with positive gains on all eight days.

In contrast, adapting only the final readout or a rank-2 LoRA readout provided little improvement beyond statistics-only alignment. This suggests that the dominant cross-day mismatch was not confined to the output mapping. Together with the contingency-reversal results, these findings support interface-specific adaptation: output-side writes compensate for changed action mappings, whereas observation-side recalibration can address shifts in the signals entering otherwise reusable computation.

5 Discussion and Limitations

Our results suggest that the usefulness of a learned representation depends not only on what can be decoded from it, but also on the learning dynamics it induces when subsequently modified. In our framework, query similarity determines how a local write propagates across states: similar queries share corrections, whereas separation protects states requiring incompatible updates. This motivates plasticity compatibility as a relational property between a representation and the update rule acting on it, rather than an intrinsic property of the representation alone. The update rule determines what is written; the learned representation determines where that write is expressed.

A complementary implication is that adaptation need not occur throughout a model when the underlying computation remains reusable. Contingency reversal could be addressed through an output-side writable state, whereas cross-day neural drift benefited from recalibration before the frozen recurrent dynamics; adapting only the final readout provided limited benefit in the latter setting. These results support an interface-specific view of adaptation: identify where deployment conditions break the correspondence with reusable computation, and modify a small state at that interface. Such writable interfaces do not replace full fine-tuning in general, but provide a controlled alternative when persistent computation should be preserved.

Several limitations remain. Our neural experiments are retrospective decoder analyses and do not establish that the proposed plasticity rule is implemented biologically or that the same behavior would hold in closed-loop deployment. Moreover, the observation-side adapter uses gradient-derived eligibility and direction-resolved feedback provides stronger supervision than binary reward. Finally, our experiments consider known classes of output- or observation-side shift. An important next step is to infer where a deployment mismatch occurs and select the appropriate writable interface online.

REFERENCES

Jimmy Ba, Geoffrey E Hinton, Volodymyr Mnih, Joel Z Leibo, and Catalin Ionescu. Using fast weights to attend to the recent past. *Advances in neural information processing systems*, 29, 2016.

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. In *Advances in Neural Information Processing Systems*, volume 38, pp. 113506–113543, 2025. doi: 10.52202/085713-3786.

Guillaume Bellec, Franz Scherr, Anand Subramoney, Elias Hajek, Darjan Salaj, Robert Legenstein, and Wolfgang Maass. A solution to the learning dilemma for recurrent networks of spiking neurons. *Nature Communications*, 11:3625, 2020. doi: 10.1038/s41467-020-17236-y.

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. doi: 10.1038/s41551-020-0542-9.

Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, Jing Yi, Weilin Zhao, Xiaozhi Wang, Zhiyuan Liu, Hai-Tao Zheng, Jianfei Chen, Yang Liu, Jie Tang, Juanzi Li, and Maosong Sun. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature Machine Intelligence*, 5(3):220–235, 2023. doi: 10.1038/s42256-023-00626-4.

Shibhansh Dohare, J. Fernando Hernandez-Garcia, Qingfeng Lan, Parash Rahman, A. Rupam Mahmood, and Richard S. Sutton. Loss of plasticity in deep continual learning. *Nature*, 632(8026): 768–774, 2024. doi: 10.1038/s41586-024-07711-7.

Yan Duan, John Schulman, Xi Chen, Peter L. Bartlett, Ilya Sutskever, and Pieter Abbeel. RL^2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pp. 1126–1135. PMLR, 2017.

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. doi: 10.1038/s41593-019-0555-4.

Wulfram Gerstner, Marco Lehmann, Vasiliki Liakoni, Dane Corneil, and Johanni Brea. Eligibility traces and plasticity on behavioral time scales: Experimental support of neohebbian three-factor learning rules. *Frontiers in Neural Circuits*, 12:53, 2018. doi: 10.3389/fncir.2018.00053.

Emmanouil Giannakakis, Sina Khajehabdollahi, and Anna Levina. Network bottlenecks and task structure control the evolution of interpretable learning rules in a foraging agent. *Artificial Life*, pp. 177–194, 2025. doi: 10.1162/artl_a_00458.

Jake Grigsby, Linxi Fan, and Yuke Zhu. Amago: Scalable in-context reinforcement learning for adaptive agents. In *International Conference on Learning Representations*, volume 2024, pp. 26919–26952, 2024.

Timothy M. Hospedales, Antreas Antoniou, Paul Micaelli, and Amos J. Storkey. Meta-learning in neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44 (9):5149–5169, 2022. doi: 10.1109/TPAMI.2021.3079209.

Neil Houlsby, Andrei Giurgica-Tiron, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2790–2799, 2019.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Yuhang Hu, Boyuan Chen, and Hod Lipson. Egocentric visual self-modeling for autonomous robot dynamics prediction and adaptation. *npj Robotics*, 3(1):14, 2025. doi: 10.1038/s44182-025-00031-6.

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, Leigh R. Hochberg, Lee E. Miller, and Chethan Pandarinath. Stabilizing brain–computer interfaces through alignment of latent dynamics. *Nature Communications*, 16(1):4662, 2025. doi: 10.1038/s41467-025-59652-y.

Jae-Hyun Kim, Kayvon Daie, and Nuo Li. A combinatorial neural code for long-term motor memory. *Nature*, 637(8046):663–672, 2025. doi: 10.1038/s41586-024-08193-3.

Louis Kirsch, James Harrison, Jascha Sohl-Dickstein, and Luke Metz. General-purpose in-context learning by meta-learning transformers. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Michael Laskin, Luyu Wang, Junhyuk Oh, Emilio Parisotto, Stephen Spencer, Richie Steigerwald, DJ Strouse, Steven Hansen, Angelos Filos, Ethan Brooks, et al. In-context reinforcement learning with algorithm distillation. In *International Conference on Learning Representations*, 2023.

Jack Lindsey and Ashok Litwin-Kumar. Learning to learn with feedback and local plasticity. In *Advances in Neural Information Processing Systems*, volume 33, pp. 21213–21223, 2020.

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.

Thomas Miconi, Kenneth Stanley, and Jeff Clune. Differentiable plasticity: training plastic neural networks with backpropagation. In *International conference on machine learning*, pp. 3559–3568. PMLR, 2018.

Thomas Miconi, Aditya Rawal, Jeff Clune, and Kenneth O Stanley. Backpropamine: training self-modifying neural networks with differentiable neuromodulated plasticity. In *International Conference on Learning Representations*, 2019.

Elias Najarro and Sebastian Risi. Meta-learning through hebbian plasticity in random networks. In *Advances in Neural Information Processing Systems*, volume 33, pp. 20719–20731, 2020.

Alexander Nikulin, Ilya Zisman, Alexey Zemtsov, and Vladislav Kurenkov. XLand-100B: A large-scale multi-task dataset for in-context reinforcement learning. In *International Conference on Learning Representations*, 2025.

Amy L. Orsborn, Helene G. Moorman, Simon A. Overduin, Maryam M. Shanechi, Dragan F. Dimitrov, and Jose M. Carmena. Closed-loop decoder adaptation shapes neural plasticity for skillful neuroprosthetic control. *Neuron*, 82(6):1380–1393, 2014. doi: 10.1016/j.neuron.2014.04.048.

Aniruddh Raghu, Maithra Raghu, Samy Bengio, and Oriol Vinyals. Rapid learning or feature reuse? towards understanding the effectiveness of MAML. In *International Conference on Learning Representations*, 2020.

Imanol Schlag, Kazuki Irie, and Juergen Schmidhuber. Linear transformers are secretly fast weight programmers. In *International Conference on Machine Learning*, pp. 9355–9366. PMLR, 2021.

Navid Shervani-Tabar and Robert Rosenbaum. Meta-learning biologically plausible plasticity rules with random feedback pathways. *Nature Communications*, 14:1805, 2023. doi: 10.1038/s41467-023-37562-1.

Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, Tatsunori Hashimoto, and Carlos Guestrin. Learning to (learn at test time): RNNs with expressive hidden states. *arXiv preprint arXiv:2407.04620*, 2024. doi: 10.48550/arXiv.2407.04620.

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*, 7:13749, 2016. doi: 10.1038/ncomms13749.

Danil Tyulmankov, Guangyu Robert Yang, and LF Abbott. Meta-learning synaptic plasticity and memory addressing for continual familiarity detection. *Neuron*, 110(3):544–557, 2022.

Jane X. Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z. Leibo, Remi Munos, Charles Blundell, Dharshan Kumaran, and Matthew Botvinick. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*, 2016.

Ilya Zisman, Vladislav Kurenkov, Alexander Nikulin, Viacheslav Sinii, and Sergey Kolesnikov. Emergence of in-context reinforcement learning from noise distillation. In *International Conference on Machine Learning*, 2024.

AI USE STATEMENT

In this work, we used generative AI tools to obtain feedback on the framing and presentation of the research, explore relevant literature, and assist with figure planning and manuscript editing. The authors reviewed and revised all AI-assisted material and independently verified the references and scientific claims included in the paper. We take full responsibility for the final content of this work, including any text or artifacts produced with the aid of generative AI.

A Baselines Details

A.1 Direction-Resolved (One-Hot) Feedback

The primary adaptation protocol uses scalar binary correctness feedback. We also evaluate a direction-resolved condition to isolate the effect of feedback identifiability from the effect of the local plasticity rule. In this condition the environment returns the intended action y_n as a one-hot target,

$$\begin{aligned} q_n &= \mathrm{onehot}(y_n) \in \mathbb{R}^A, \\ \delta_n^{\mathrm{dir}} &= q_n - \pi_n. \end{aligned} \tag{18}$$

The fast state, query, slow readout, and persistent parameters are unchanged from the binary-feedback method. Only the action-side teaching signal is replaced:

$$\begin{aligned} B_{n+1} &= \lambda_B B_n + \eta_B \, \delta_n^{\mathrm{dir}} z_n^\top, \\ \ell_n &= \ell_n^{\mathrm{slow}} + g B_n z_n, \\ \pi_n &= \mathrm{softmax}(\ell_n). \end{aligned} \tag{19}$$

This update is the action–query outer product of the full softmax error. It is equivalent to a single-step gradient update of the linear readout with respect to the target cross-entropy, except that it is implemented as a bounded, resettable evaluation-time fast state rather than as persistent readout optimization. No GRU parameter, query projection, slow readout, optimizer momentum, or persistent adapter is changed.

This protocol is retrospective decoder adaptation: the sampled action is scored against the recorded target to construct binary feedback, but it does not alter the animal's subsequent recorded neural activity. Consequently, the experiment tests whether a small temporary interface can recover decoding under session drift, not closed-loop animal–decoder co-adaptation.

A.2 Competitive Baseline Implementations

This section specifies the baseline implementations used in the arbitrary-permutation session experiment. The purpose is to distinguish update capacity from feedback information. All methods start from the same outer-trained trial-local GRU checkpoint, see the same target trials in chronological order, use the same support values $K \in \{0, 5, 10, 20, 50\}$, and use the same sampled support actions and random uniforms. During query trials, actions are greedy and no further update is made. At the beginning of every target session, all temporary states are reset. The target action is used only by the simulated environment to return the binary outcome

$$r_t = \begin{cases} +1, & a_t = a_t^\star, \\ -1, & a_t \neq a_t^\star, \end{cases} \tag{20}$$

and by the offline scorer. It is not passed to a binary-feedback update.

A.2.1 No-adaptation and local-plasticity controls

Frozen decoder. The frozen condition evaluates the source checkpoint without changing any parameter or evaluation state. It measures the performance cost of the held-out action permutation and supplies the $K = 0$ reference. It has no online teaching signal.

Hebbian writer. The Hebbian control writes a chosen-action outer product without outcome sign, which is the standard pre–post co-activity form of a local plasticity rule (Gerstner et al., 2018):

$$B_{t+1} = \Pi_{\mathcal{B}} \left(\lambda_B B_t + \eta \, \mathrm{onehot}(a_t) z_t^\top \right). \tag{21}$$

It receives the chosen action but ignores whether that action was correct. Thus its update capacity and fast-state size are matched to the signed writers, while its feedback resolution is deliberately lower.

A.2.2 BINARY-FEEDBACK GRADIENT BASELINES

The following methods receive exactly the same (a_t, r_t) pair as the local writers, but use a gradient update and optimizer state. For all of them, the binary feedback objective is

$$\mathcal{L}_t^{\text{bin}} = -r_t \log \pi_t(a_t) + \frac{1-\lambda_B}{2\eta}\|\vartheta_t - \vartheta_0\|_2^2, \tag{22}$$

where ϑ_0 is the source checkpoint and the second term implements the same retention convention as the fast-state baselines. For $r_t = -1$, the likelihood term suppresses the selected action but does not identify one of the other $A-1$ actions as correct. This is the central feedback- identifiability difference between these controls and a correct-label upper bound.

Adaptive readout. Only the slow action-readout matrix is trainable:

$$\ell_t = W_{\text{slow,t}} z_t, \qquad W_{\text{slow,t+1}} = W_{\text{slow,t}} - \alpha \nabla_{W_{\text{slow}}} \mathcal{L}_t^{\text{bin}}. \tag{23}$$

The GRU, query projection, and input encoder remain frozen. This is the smallest gradient baseline and is conceptually close to fitting a task head over a reusable representation, as emphasized in analyses of rapid adaptation and feature reuse (Finn et al., 2017; Raghu et al., 2020; Hospedales et al., 2022).

Persistent LoRA. The LoRA control factorizes the residual as $R_t = U_t V_t$ with rank two:

$$\ell_t = \ell_t^{\text{slow}} + g U_t V_t z_t, \qquad (U_{t+1}, V_{t+1}) = (U_t, V_t) - \alpha \nabla_{U,V} \mathcal{L}_t^{\text{bin}}. \tag{24}$$

The base GRU, query projection, and slow readout remain frozen. This follows the low-rank adaptation principle of LoRA (Hu et al., 2022); its persistent low-rank factors are distinct from the episode-local full-rank B_t.

Full persistent fine-tuning. All persistent parameters are trainable:

$$\vartheta_t = \{\Theta_{\text{enc}}, \Theta_{\text{rec}}, \Theta_q, W_{\text{slow}}\}, \qquad \vartheta_{t+1} = \vartheta_t - \alpha \nabla_\vartheta \mathcal{L}_t^{\text{bin}}. \tag{25}$$

This is a high-capacity comparison to inner-loop gradient adaptation, not a parameter-efficient baseline. It can change recurrent dynamics and query geometry, so it tests a different hypothesis from our method: that the representation itself should be retuned during the target session. The comparison is intentionally feedback matched and does not imply that ordinary fine-tuning is universally ineffective.

B DATASET DESCRIPTIONS

B.1 SYNTHETIC STRUCTURED REMAPPING DATA

The main synthetic demonstration uses a deliberately minimal two-session protocol. One source session with 200 trials is used for outer-loop learning. Another session with 200 trials is generated with a new fixed action mapping and is used only for inner-loop adaptation and evaluation.

Let the stimulus set and action set both be $\mathcal{S} = \mathcal{A} = \{0, 1, 2, 3\}$. The source mapping is the identity,

$$\rho_{\text{src}}(s) = s. \tag{26}$$

We generate exactly $N_{\text{src}} = 200$ trials. On each trial, a stimulus s_n is sampled independently and uniformly from $\mathcal{S}$. The source label is therefore $y_n^{\text{src}} = \rho_{\text{src}}(s_n)$. Each trial has $T = 8$ ordered time bins and six observation channels. The first four channels encode the stimulus. During the first three bins, channel k contains a noisy one-hot cue:

$$x_{n,\tau,k} = \mathbf{1}[k = s_n] + \epsilon_{n,\tau,k}, \qquad \tau \in \{1, 2, 3\}, \tag{27}$$

where

$$\epsilon_{n,\tau,k} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma_{\text{obs}}^2), \qquad \sigma_{\text{obs}} = 0.1. \tag{28}$$

For bins $\tau \in \{4, 5, 6, 7, 8\}$, the four stimulus channels contain only the corresponding Gaussian noise. The fifth channel is a cue-phase indicator $x_{n,\tau,4} = \mathbf{1}[\tau \leq 3]$, and the sixth channel marks the terminal decision bin $x_{n,\tau,5} = \mathbf{1}[\tau = 8]$. Thus each source trial is an array $X_n = (x_{n,1}, \ldots, x_{n,8}) \in \mathbb{R}^{8\times 6}$.

Remapped target session. We independently generate a second session with $N_{\mathrm{tgt}} = 200$ trials. Stimuli are sampled in the same way and have the same observation distribution, but the action mapping is replaced by a new fixed permutation:

$$\rho_{\mathrm{tgt}} \neq \rho_{\mathrm{src}}, \qquad y_n^{\mathrm{tgt}} = \rho_{\mathrm{tgt}}(s_n). \tag{29}$$

The new permutation is sampled once before the target session and remains unchanged for all 200 target trials. It is not supplied to the model. The target observations have exactly the same shape and cue-generation process as the source observations. Only the hidden stimulus-to-action mapping changes. Consequently, the target session isolates output remapping rather than observation drift.

B.2 DANDI 001188

We evaluated the decoder on the publicly available DANDI 001188 dataset, which contains calcium-imaging recordings from the tactile delayed-lick task reported by Kim, Daie, and Li Kim et al. (2025). The task contains binary left/right responses following a pole-sample period. We constructed source-target pairs from recordings of the same animal and imaging field of view (FOV) acquired under different expert contexts. The fixed pair manifest, including animal, FOV, source date, and target date, is provided. Development pairs were used to establish the source-training and adaptation protocol; held-out pairs were not used for selecting preprocessing, architecture, or adaptation hyperparameters

B.2.1 Trial labels and quality control

The NWB trial tables contain six mutually exclusive outcome indicators: `HitL`, `HitR`, `ErrL`, `ErrR`, `NoLickL`, and `NoLickR`. The instructed action $y_i \in \{0, 1\}$ is determined by the side associated with the active indicator,

$$y_i = \begin{cases} 0, & \text{HitL, ErrL, or NoLickL,} \\ 1, & \text{HitR, ErrR, or NoLickR.} \end{cases} \tag{30}$$

The emitted response is stored separately and is not used as a substitute for the instructed action. We retained trials marked `GoodTrials`, excluded trials marked `LickEarly`, and required exactly one valid instructed-side indicator. The primary benchmark further retained rewarded trials (`HitL` or `HitR`), matching the established analysis protocol. Error and no-lick trials remain available to the loader for quality control but are not silently included in the primary training set.

The target action label is an offline analysis variable. In the binary-feedback condition, it is used only by the retrospective environment to determine whether the decoder's sampled action was correct. It is not passed to the reward-modulated writer. Conditions that receive the correct action directly are reported as label-supervised upper bounds.

B.3 Cross-day neural decoding

B.3.1 Dataset and behavioral task

We used twelve recording sessions from the publicly available center-out reaching dataset (Karpowicz et al., 2025). In each trial, the animal moved a cursor toward one of eight radial targets while intracortical spiking activity and cursor kinematics were recorded. We formulated the retrospective decoding task as predicting the intended target direction from post-go-cue neural activity. The decoded output was therefore an eight-way direction class; continuous cursor velocity was not decoded. This analysis evaluates offline cross-day decoder adaptation and should not be interpreted as a closed-loop BCI experiment.

B.3.2 Trial inclusion and Neural preprocessing

The primary analysis included only rewarded trials, identified by the XDS result code `R`. Thus, trials with failed, aborted, or invalid behavioral outcomes were excluded before model training and evaluation.

We retained the 92 electrode identities present in all twelve sessions and preserved a fixed channel order across days. Spike counts were aligned to the go cue, and only activity in the interval $[0, 800)$ ms after the go cue was analyzed. The original 1-ms spike counts were aggregated into non-overlapping

20-ms firing-rate bins, producing a sequence $X_i \in \mathbb{R}^{40\times92}$for trial i. Firing rates were smoothed independently within each trial using a symmetric Gaussian kernel with $\sigma = 50$ ms. Smoothing was never performed across trial boundaries. All normalization statistics used by the persistent decoder were estimated from the source session only.

B.3.3 CHRONOLOGICAL DATA SPLIT

Sessions were divided chronologically. Day 1 was used for persistent decoder training. Days 2–4 were used for development and hyperparameter selection. Days 5–12 were treated as historical holdout sessions and were not used to select the projection dimension, learning rate, retention, alignment strength, or feedback protocol. Because these sessions had been inspected during earlier exploratory analyses, we refer to them as historical holdout rather than newly blinded test data.

For each target session, the first $K \in \{0, 1, 5, 10, 20, 50\}$ chronological trials were used as adaptation support. Performance was evaluated on a common tail beginning at trial 50, ensuring that different support budgets were compared on the same evaluation trials.