BRAIN-CONDITIONED ACTION POLICIES FOR NEURAL MOTOR DECODING

Anonymous authors
Paper under double-blind review

ABSTRACT

Motor brain-computer interfaces (BCIs) aim to decode motor intention, enabling people with paralysis to control external devices. Neural motor decoding typically learns task-specific mappings from neural activity to kinematics, yet remains constrained by scarce paired neural–action data. We propose BrainVLA, a framework that enables neural motor decoding by drawing on a pretrained vision-language-action (VLA) model through language-mediated alignment. BrainVLA mitigates reliance on scarce paired neural–action data by leveraging VLA policies. We first construct VLA-compatible datasets including paired neural activity, action signals, language instructions, and rendered visual observations. Then, we adapt the OpenVLA-OFT policy to the target action spaces through LoRA fine-tuning. To establish an effective interface through which neural activity can convey motor intention to adapted VLA policies and guide action generation, we train a neural encoder via neural–language alignment, using language representations as semantic targets to capture latent motor intent from neural activity. The resulting neural representations serve as an endogenous intention signal to guide VLA policies to generate executable actions, while visual observations provide complementary information about the evolving task state. BrainVLA is evaluated on two neural motor datasets with different action dimensionalities using causal rollout decoding. It outperforms the evaluated baselines in cross-session decoding R^2 and task success rate, while demonstrating high training data efficiency. These results establish a route for neural motor decoding to draw on large-scale robotic priors through brain-conditioned VLA policies.

1 INTRODUCTION

Decoding motor activity from neural recordings is a central problem in brain–computer interfaces, with the potential to enable paralyzed patients to control external devices through attempted movement Lee et al. (2025); Willsey et al. (2025). Conventional neural decoding research has largely pursued two directions: developing increasingly capable model architectures to improve task-specific mapping from neural activity to kinematic variables Karpowicz et al. (2025); Azabou et al. (2025); Le et al. (2026); Jiang et al. (2026), and scaling training across multiple neural datasets to learn more generalizable neural representations that can be adapted to downstream tasks Zhang et al. (2025); Ye et al. (2026). Nevertheless, neural motor decoding remains constrained by the limited availability of paired neural-action data. These limitations motivate us to ask: can neural motor decoding benefit from action priors within models trained on large-scale action data?

Vision-language-action (VLA) models provide a promising avenue for investigating this question, because they learn action policies from large-scale robotic demonstrations and vision-language corpora and have demonstrated strong performance across diverse robotic tasks, such as RT-2 Brohan et al. (2023), OpenVLA Kim et al. (2024; 2025), and its variant OpenVLA-OFT Kim et al. (2025). However, harnessing these visuomotor policies and action priors for neural motor decoding remains underexplored. A key challenge is establishing an effective interface through which neural activity can convey motor intention to VLA policies and guide action generation. Recent studies have revealed geometric similarities between neural activity and language representations learned by language models Goldstein et al. (2024); Mischler et al. (2024). These findings motivate us to explore language as a semantic bridge: aligning neural and language representations may help capture latent motor intent and enable neural activity to guide VLA policies through their original language

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.

interface. As an additional benefit of this integration, visual observations further ground the policy in the initial and evolving state.

Here, we introduce **Brain**-conditioned action policies based on the **VLA** model (**BrainVLA**), a framework that enables neural activity to condition on an adapted VLA model for motor decoding through language-mediated alignment. We first construct VLA-compatible datasets by augmenting existing paired neural-action recordings with rendered observations and generated language instructions, constructing image-language-action examples. We use these examples to adapt OpenVLA-OFT Kim et al. (2025) with LoRA Hu et al. (2021) to establish a language-conditioned reference policy for the target motor task. To enable neural activity to guide the policy, we train a neural encoder to align its outputs with language representations of the corresponding motor intent. This alignment provides a semantic target for learning intent-related neural representations. The adapted policy then generates actions from neural representation convey inferred motor intent, and visual observations supply complementary current task state. Together, task-specific policy adaptation and neural–language alignment allow BrainVLA to benefit neural motor decoding from pretrained robotic action priors, with the aim of improving decoding performance and training data efficiency.

We evaluate BrainVLA on two datasets with different action dimensions using causal action decoding. To assess whether decoded actions accomplish the intended tasks, we introduce task-level success criteria alongside conventional decoding R^2. Across both datasets, BrainVLA outperforms the evaluated baselines in cross-session decoding R^2 and task success rate, while demonstrating high training data efficiency. Further analyses reveal complementary modality contributions, with vision primarily supporting spatial guidance, whereas neural activity contributes more to distal motor control. In summary, the main contributions include:

- We introduce **BrainVLA**, a framework that decodes motor intent from neural activity, drawing on the VLA model. This provides a new strategy that leverages large-scale VLA policies for neural motor decoding.
- We develop a language-mediated approach that aligns the neural representation with language representations to enable neural signals to guide VLA policies. This alignment provides a semantic target for capturing latent motor intention from neural activity.
- We evaluate BrainVLA on two datasets spanning different action dimensions under a causal rollout decoding protocol. It supports zero-shot transfer to held-out sessions and remains effective in low-data regimes.

2 Related Work

Models for motor neural decoding Recent motor decoding models have been developed by learning general neural representations. NDT2 Ye et al. (2023) and NDT3 Ye et al. (2026) use Transformer-based pretraining to capture shared structure in large-scale neural recordings, while POYO Azabou et al. (2023) and POYO+ Azabou et al. (2025) employ spike-level tokenization and cross-attention to accommodate heterogeneous neural populations. NEDS jointly models neural and behavioral signals Zhang et al. (2025), MINT leverages the geometry of population dynamics Perkins et al. (2025), and NoMAD Karpowicz et al. (2025), SPINT Le et al. (2026), and MANDT Jiang et al. (2026) address cross-session variability and neural nonstationarity through latent alignment, permutation-invariant representations, or context-invariant dynamics. Collectively, these approaches primarily improve decoding by modeling, scaling, or stabilizing representations on the neural side, with behavioral variables serving mainly as decoding targets. However, they remain constrained by the limited availability of paired neural-action data. Our work mitigates reliance on scarce paired neural–action data by leveraging VLA policies.

Vision-language-action policies Vision-language-action (VLA) models condition action generation on visual observations and language instructions by integrating visual and linguistic representations within a pretrained multimodal policy. RT-2 demonstrated that knowledge acquired by large vision-language models can transfer to robotic control Brohan et al. (2023), while OpenVLA provided an open-source VLA pretrained on diverse robot demonstrations and supporting parameter-efficient adaptation Kim et al. (2024). Related generalist policies such as Octo emphasize efficient adaptation across embodiments Team et al. (2024), while π_0 introduces flow matching for continuous, dexterous action generation Black et al. (2026). Building on OpenVLA, OpenVLA-OFT

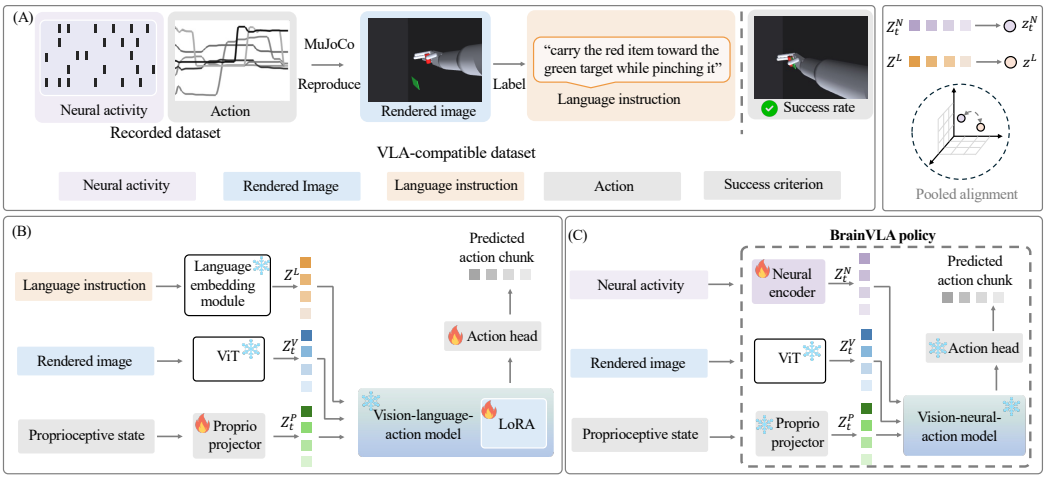

Figure 1: Overview of BrainVLA. (A) The construction of the VLA-compatible dataset, with MuJoCo-rendered observations, language instructions, and task-specific success criteria. (B) The task-specific VLA adaptation, with language, vision, and proprioception, while task-specific modules and LoRA parameters are optimized. (C) Train a neural encoder to map neural activity into the VLA conditioning space through pooled neural-language alignment.

improves and stabilizes downstream adaptation Kim et al. (2025). Given the limitations of neural data and advances in pretrained action models, we explore VLA policies as a possibility for neural motor decoding. We therefore adopt OpenVLA-OFT as our policy backbone, leveraging its efficient and publicly reproducible fine-tuning framework to adapt a VLA policy to neural motor tasks.

Cross-modal neural–semantic alignment Cross-modal alignment provides a principled way to bridge the representational gap between neural activity and other modalities. CEBRA demonstrates that behavioral and temporal information can structure neural latent representations through contrastive learning Schneider et al. (2023). Recent studies further reveal substantial correspondence between neural representations during language processing and representations learned by language models, including shared embedding geometry and contextual processing hierarchies Goldstein et al. (2024); Mischler et al. (2024), with model scale and context further influencing brain-language-model alignment Raugel et al. (2026); Gao et al. (2025). Further, DeWave explicitly aligns EEG representations with pretrained language models for text decoding Duan et al. (2023), while BIT uses contrastive alignment between neural representations and text embeddings for brain-to-text decoding Zhang et al. (2026). In our work, we align neural representations with language representations to capture the latent motor intention from neural activity and enable neural activity to guide the VLA policy to generate executable actions.

3 Methods

3.1 Problem Formulation

We consider causal rollout neural motor decoding as a mapping problem $f_\theta : \mathbf{N}_t \to \hat{\mathbf{A}}_t$. At time t, the decoder observes a causal neural history $\mathbf{N}_t = \mathbf{N}_{[t-\tau,t]}$, the goal is to predict an action chunk of horizon H: $\hat{\mathbf{A}}_t = [\hat{\mathbf{a}}_t, \ldots, \hat{\mathbf{a}}_{t+H-1}]$, using only neural information available up to the current time. Conventional neural decoders directly learn a task-specific mapping $f_\theta : \mathbf{N} \to \hat{\mathbf{A}}$. In contrast, BrainVLA formulates neural decoding as conditioning a VLA policy on neural activity, which can utilize the action prior from the large-scale VLA policy. Let g_ϕ denote a task-adapted VLA policy that predicts actions from visual observations $\mathbf{o}_t$, proprioceptive state $\mathbf{p}_t$, and a conditioning representation $\mathbf{Z}^L$: $\hat{\mathbf{A}}_t = g_\phi\left(\mathbf{o}_t, \mathbf{p}_t, \mathbf{Z}^L\right)$. During VLA adaptation, the condition signal is provided by the language embedding $\mathbf{Z}^L$ describing the intended action. BrainVLA subsequently

learns a neural encoder f_ψ that maps causal neural activity to a representation $\mathbf{Z}_t^N = f_\psi(\mathbf{N}_t)$ by neural-language alignment, $\mathbf{Z}_t^N$ and $\mathbf{Z}^L$ are pooled into single-vector representations $\mathbf{z}_t^N$ and $\mathbf{z}^L$, respectively. The alignment enables the neural representation $\mathbf{z}_t^N$ to capture the latent motor intention of corresponding language representations $\mathbf{Z}^L$. At deployment, language is no longer required; the neural representation directly replaces the language conditioning: $\hat{\mathbf{A}}_t = g_\phi\left(\mathbf{o}_t, \mathbf{p}_t, \mathbf{Z}_t^N\right)$. Predicted actions are executed in MuJoCo to update the task state and generate the subsequent visual observation $\mathbf{o}_{t+\Delta}$. Meanwhile, the causal neural window advances from $\mathbf{N}_{[t-\tau,t]}$ to $\mathbf{N}_{[t+\Delta-\tau,t+\Delta]}$. This yields a causal rollout decoding process in which neural activity provides the intended action signal, while evolving visual and proprioceptive observations provide the current state.

3.2 VLA-Compatible Dataset Construction

We evaluate BrainVLA on two neural motor datasets with distinct embodiments and action spaces: Dataset 1, containing human neural activity paired with 7-DoF robotic-arm actions Collinger et al. (2013); Wodlinger et al. (2015); Flesher et al. (2021), and Dataset 2, containing non-human primate neural activity paired with 2-DoF finger-control actions Nason et al. (2021). Both datasets provide synchronized neural and behavioral signals, but lack the visual and language modalities required by VLA policies. To bridge the mismatch between existing neural datasets and VLA training requirements, we construct the VLA-compatible multimodal dataset. As shown in Fig. 1(A), we replay the recorded action signals in MuJoCo and render the corresponding task observations from a fixed third-person RGB camera. Each trial is additionally assigned a language instruction ℓ describing the intended motor behavior represented by the reconstructed vision observations. The resulting RLDS dataset contains synchronized tuples $(\mathbf{N}_t,\, \mathbf{o}_t,\, \ell,\, \mathbf{a}_t)$, where $\mathbf{N}_t$ denotes neural activity, $\mathbf{o}_t$ the rendered observation, ℓ the language instruction, and $\mathbf{a}_t$ the action target. We additionally define dataset-specific task-success criteria to complement conventional trajectory metrics R^2. These criteria provide a behavior-level measure of whether the decoded actions accomplish the intended motor task. These measures quantify success under kinematic reconstruction based on predicted action. For D1, success is determined using task-dependent geometric and hand-configuration criteria in MuJoCo. For D2, success is evaluated from endpoint error after integrating the predicted finger velocities. The detailed definitions are provided in Sec. A.4.

3.3 Framework structure

Adapted VLA policy. The robotic demonstrations and vision-language corpora used to pretrain the backbone OpenVLA-OFT policy are different from our neural motor tasks. We first adapt the OpenVLA-OFT policy Kim et al. (2025) to each neural motor task using the constructed VLA-compatible data, as illustrated in Fig. 1(B). Given a rendered observation $\mathbf{o}_t$, language instruction ℓ, and proprioceptive state $\mathbf{p}_t$, the frozen visual and language modules produce the corresponding visual $\mathbf{Z}_t^V$ and language representations $\mathbf{Z}^L$, while a learnable proprioception projector maps $\mathbf{p}_t$ into the policy embedding space $\mathbf{Z}_t^P$. These representations are then processed by the pretrained OpenVLA-OFT backbone. During adaptation, the pretrained backbone remains frozen, while LoRA adapters Hu et al. (2021), the task-specific action head, and the proprioception projector are optimized because of the diverse action dimensions. This parameter-efficient adaptation accommodates the distinct state and action spaces of the two motor tasks without updating the full VLA backbone. The resulting policy predicts an action chunk $\hat{\mathbf{A}}_t = g_\phi\left(\mathbf{o}_t,\, \ell,\, \mathbf{p}_t\right)$, where ϕ denotes the learnable LoRA parameters, action head, and proprioception projector. This stage establishes a task-adapted visuomotor policy that is subsequently conditioned on neural representations in BrainVLA.

Neural encoder. Following POYO Azabou et al. (2023), we represent neural activity as an asynchronous sequence of spike events, where each spike token encodes its event time and unit identity. This formulation accommodates recordings with variable numbers of spikes and neural units within a shared architecture. A set of learned latent queries $\mathbf{Q}$ first attends to the spike sequence through cross-attention, compressing the variable-length neural input into a fixed-size latent representation. The resulting latent tokens are then refined through multiple self-attention blocks to capture interactions across the neural population and temporal context. Formally, we use the final latent representation $\mathbf{Z}_t^N$ to represent the causal neural window $\mathbf{N}_{[t-\tau,t]}$. This representation subsequently serves as the neural conditioning input to the downstream VLA policy.

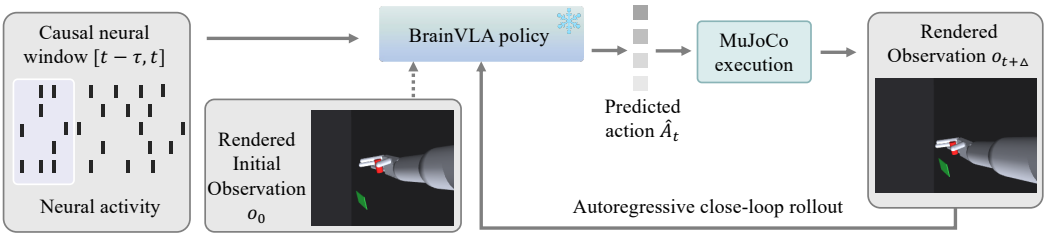

Figure 2: BrainVLA causal rollout protocol. At each step, a causal neural window and the current MuJoCo-rendered observation are used to predict action $\hat{A}_t$. Executing $\hat{A}_t$ updates the environment and produces the next rendered observation $o_{t+\Delta}$, which is fed back to BrainVLA while the neural window advances. Thus, inference uses only past neural activity and self-evolving visual observations generated by the model's own actions.

Neural-language alignment. To learn the intent-related neural representation $\mathbf{Z}_t^N$ and enable it to guide VLA policies, we train the neural encoder by aligning its representation with language representations of the corresponding motor intent, as illustrated in Fig. 1(C). Because language representations usually contain different numbers of tokens, we perform alignment in a pooled embedding space. Specifically, the neural representation $\mathbf{Z}_t^N$ is mean-pooled over latent tokens to obtain $\mathbf{z}_t^N$, while the language token representation $\mathbf{Z}^L$ is obtained from the instruction using the frozen OpenVLA tokenizer and token-embedding module Kim et al. (2025) and mean-pooled over valid instruction tokens to obtain $\mathbf{z}^L$. Both pooled representations are ℓ_2-normalized before alignment Zhang et al. (2026). We optimize a symmetric multi-positive contrastive objective over each mini-batch Khosla et al. (2021). The cosine similarity between neural example i and language example j is $S_{ij} = \left(\hat{\mathbf{z}}_i^N\right)^\top \hat{\mathbf{z}}_j^L$, where $\hat{\mathbf{z}}^N$ and $\hat{\mathbf{z}}^L$ denote the normalized pooled embeddings. Samples sharing the same instruction are treated as positives, preventing neural examples associated with the same intended behavior from being incorrectly treated as negatives. The alignment objective is computed symmetrically in both neural-to-language and language-to-neural directions:

$$\mathcal{L}_{\text{align}} = \frac{1}{2}\left(\mathcal{L}_{N\to L} + \mathcal{L}_{L\to N}\right). \tag{1}$$

The detailed parameters are in Sec. B.3. This objective encourages neural representation capture the latent motor intention of the corresponding language instruction from neural activity.

Brain-conditioned action policy. After neural-language alignment, the neural representation $\mathbf{Z}_t^N$ replaces the language conditioning $\mathbf{Z}^L$ of the task-adapted VLA policy. At time t, the causal neural window $\mathbf{N}_{[t-\tau,t]}$ is encoded into a neural conditioning representation $\mathbf{Z}_t^N$, which is provided to the policy together with the current visual observation $\mathbf{o}_t$ and proprioceptive state $\mathbf{p}_t$. The resulting brain-conditioned policy predicts an action chunk $\hat{\mathbf{A}}_t = g_\phi\left(\mathbf{o}_t, \mathbf{p}_t, \mathbf{Z}_t^N\right)$, without requiring a language instruction at inference time. The predicted actions are executed in MuJoCo to update the environment $o_{t+\Delta}$ and produce the subsequent observation, enabling BrainVLA to operate as a neural-conditioned visuomotor policy.

3.4 Training and Inference

Training. BrainVLA is trained on the VLA-compatible RLDS dataset constructed in Sec. 3.2. For each timestep t, the model receives a causal neural window $\mathbf{N}_{[t-\tau,t]}$, the ground-truth rendered observation $\mathbf{o}_t$ stored in the dataset, and the corresponding proprioceptive state $\mathbf{p}_t$. The neural encoder maps the neural activity to a conditioning representation $\mathbf{Z}_t^N$, which replaces the language representation in the task-adapted VLA policy. The policy then predicts an action chunk $\hat{\mathbf{A}}_t$ supervised by the recorded action target $\mathbf{A}_t$. Training jointly optimizes action prediction and neural-language alignment: $\mathcal{L} = \mathcal{L}_{\text{action}} + \lambda_{\text{align}}\mathcal{L}_{\text{align}}$, with $\lambda_{\text{align}} = 0.01$ makes two losses are comparable. Starting from the task-adapted VLA, we keep the VLA backbone frozen while optimizing the neural encoder. Note that training uses observations corresponding to the recorded ground-truth trajectories each cycle rather than states evolved by the model's own predictions.

Table 1: Decoding performance comparison. The best results are highlighted in bold.

Method	D1		D2	
	$\mathbf{R^2}$	SR	$\mathbf{R^2}$	SR
Wiener filter (Glaser et al. (2020))	0.15±0.11	0.13±0.08	0.14±0.11	0.19±0.06
RNN (Glaser et al. (2020))	0.32±0.16	0.21±0.17	0.18±0.21	0.26±0.12
CycleGAN+WF (Ma et al. (2023))	-0.09±0.15	0.07±0.05	-0.09±0.10	0.12±0.03
POYO (Azabou et al. (2023))	0.14±0.22	0.34±0.14	0.12±0.18	0.24±0.09
NDT2 (Ye et al. (2023))	0.21±0.18	0.16±0.13	0.23±0.14	0.24±0.12
NoMAD (Karpowicz et al. (2025))	0.18±0.21	0.30±0.10	0.23±0.16	0.36±0.05
SPINT (Le et al. (2026))	0.19±0.12	0.06±0.06	0.05±0.09	0.15±0.06
NDT3 (Ye et al. (2026))	0.25±0.11	0.18±0.08	0.29±0.10	0.27±0.09
BrainVLA(Ours)	**0.65±0.11**	**0.77±0.10**	**0.46±0.05**	**1.00±0.00**

Causal rollout inference. At inference time, language instruction supervision is removed, and the aligned neural representation directly conditions the VLA policy, as shown in Fig. 2. At the initial step, the neural history is zero-padded and paired with the initial observation o_0. The causal neural window τ is 1-s. At each subsequent query step t, BrainVLA receives only the causal neural history $\mathbf{N}_{[t-\tau,t]}$, the current evolved observation $\mathbf{o}_t$, and the current proprioceptive state $\mathbf{p}_t$:

$$\hat{\mathbf{A}}_t = g_\phi \left(\mathbf{o}_t,\, \mathbf{p}_t,\, f_\psi(\mathbf{N}_{[t-\tau,t]}) \right). \tag{2}$$

The predicted action chunk is executed in MuJoCo, which updates the environment state and produces the next visual observation $\mathbf{o}_{t+\Delta}$ and proprioceptive state $\mathbf{p}_{t+\Delta}$. The neural window then advances by the same query interval $\mathbf{N}_{[t-\tau,t]} \rightarrow \mathbf{N}_{[t+\Delta-\tau,t+\Delta]}$, and the process repeats until the episode terminates. We use an 8-step action chunk and query every 8 steps. Unlike training, causal rollout inference does not use the ground-truth images for each query step. Instead, visual and proprioceptive inputs evolve according to BrainVLA's previously predicted actions, allowing prediction errors to affect subsequent states. This autoregressive protocol evaluates BrainVLA as a causal brain-conditioned action policy rather than as an offline neural-to-action regressor.

4 Results

4.1 Experiment setup

We split the official held-in data into approximately 80% for training and 20% for validation, while reserving the official held-out sessions exclusively for final testing; detailed splits are provided in Section A.1. BrainVLA and all reproduced baselines use the same data splits and causal rollout evaluation protocol (Fig. 2). We measure BrainVLA policy inference latency on an NVIDIA RTX PRO 6000 Blackwell Server Edition. For an 8-action prediction chunk, BrainVLA requires 0.064 s per query on D1 and 0.045 s per query on D2, respectively. Performance is evaluated using action R^2 and task success rate (SR) across held-out sessions.

4.2 BrainVLA Decoding performance

We next evaluate whether conditioning the policy on neural activity can be effective for motor decoding. Table 1 compares BrainVLA with conventional neural decoding methods on both D1 and D2. All methods are reproduced using the same train/valid/test setting as our method. Wiener and RNNGlaser et al. (2020) provide conventional supervised decoding baselines; CycleGANMa et al. (2023) and NoMADKarpowicz et al. (2025) address cross-session variability through neural representation alignment; SPINTLe et al. (2026) models variable neural populations using permutation-invariant unit representations; and NDT2Ye et al. (2023), NDT3Ye et al. (2026), and POYOAzabou et al. (2023) represent recent Transformer-based neural decoding approaches. Detailed implementations are provided in Appendix C.1. BrainVLA achieves the strongest performance of zero-shot transfer to held-out sessions among the currently evaluated methods. D2's high SR despite moderate R^2 may reflect visually guided target attainment without reproducing the reference trajectory. Fig.

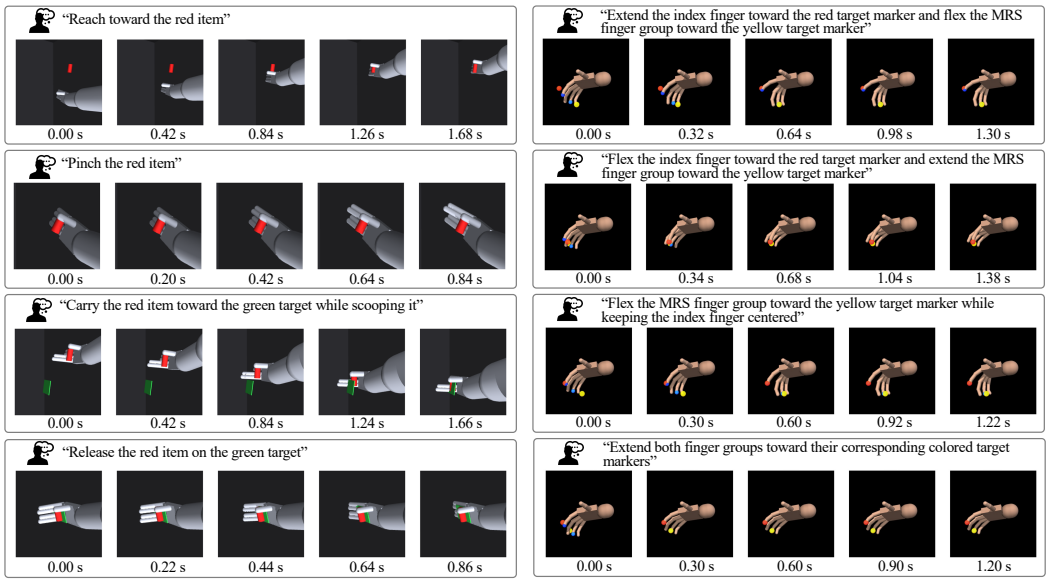

Figure 3: Representative causal BrainVLA rollouts on D1 (left panel) and D2 (right panel). Each pane shows sequential observations from one neural-conditioned episode.

3 shows D2 reaches the target by around the 3rd frame; differences in timing and subsequent motion can lower R^2 while preserving task success. The consistent gains for both datasets suggest that the proposed neural conditioning VLA policy is applicable across different action spaces and embodiments. The superior results indicate that BrainVLA has gained benefits from the VLA policy, and the language-mediated strategy is effective in enabling neural activity to guide the VLA policy.

4.3 Closed-Loop Rollout Result

Figure 3 shows representative causal BrainVLA rollouts on both datasets. Each pane corresponds to a different motor episode, with frames sampled sequentially from the autoregressive rollout. The text is the corresponding language instruction, which is only used to show motor intention. Each visual observation is rendered from the state produced by previous predictions, which reflect the accumulated behavior of the policy rather than independent frame-wise predictions. The examples qualitatively demonstrate that BrainVLA can complete coherent neural-conditioned behavior over multiple rollout steps.

4.4 Training Data Efficiency

We evaluate training data efficiency by training each model with diverse fractions of the D1 training set. As shown in Fig. 4 (A), BrainVLA consistently outperforms POYO and LSTM across data regimes in both trajectory R^2 and task success rate. BrainVLA improves steadily with increasing training data, while the baselines exhibit substantially lower performance and greater variability, particularly in the low-data regime. Notably, BrainVLA retains meaningful decoding and task-level success even when trained with only a small subset of data. These results suggest that our policy improves the data efficiency of neural motor decoding.

4.5 Ablation Studies

We evaluate BrainVLA through two complementary ablation settings. First, **component ablations** isolate the contribution of major architectural and training components. Each variant is independently retrained and reevaluated under the same data splits and causal rollout protocol. These results are summarized in Table 2. Second, **modality ablations** examine the dependence of the trained

Table 2: Component ablation. Each variant is retrained and reevaluated to assess the contributions.

Variant	Decoder	Alignment	Vision	D1		D2	
				R^2	SR	R^2	SR
BrainVLA	adapted VLA	✓	Evolving	0.65	0.77	0.46	1.00
w/o VLA adapt.	OpenVLA-7B	✓	Evolving	0.39	0.43	0.34	0.75
w/o language	adapted VLA	✗	Evolving	0.56	0.68	0.41	0.87
w/o vision	adapted VLA	✓	White	0.29	0.52	0.13	0.44
w/o VLA	MLP	✓	Evolving	0.14	0.31	0.18	0.22

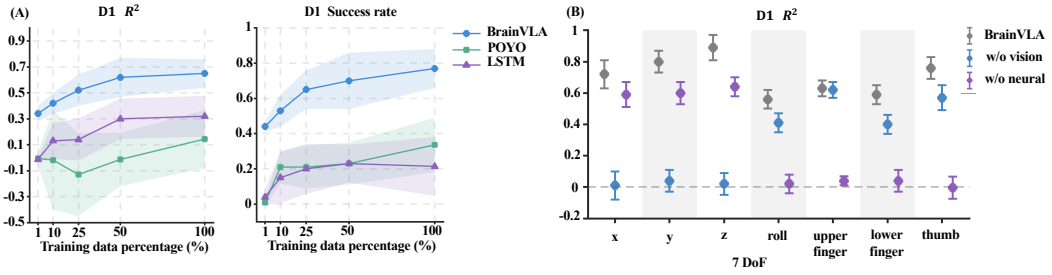

Figure 4: (A) R^2 and task success rate with scaled training data. BrainVLA consistently outperforms POYO and LSTM across data regimes and remains effective under limited neural supervision. Shaded regions indicate variability across test sessions. (B) R^2 for each of the seven action dimensions under the full model and w/o vision or neural input. Vision removal primarily degrades the translational dimensions (x, y, z), whereas neural removal more strongly affects roll and finger-related dimensions.

policy on neural activity, vision, and proprioception. The BrainVLA model is kept fixed, and input modalities are removed only at inference time. The results are reported in Table 3.

Component Ablation. (1) VLA adaptation. To evaluate the importance of task-specific VLA adaptation, we replace the adapted VLA backbone with the original OpenVLA-7B backbone, while keeping the neural encoder, action head, and proprioception projector trainable. The resulting degradation indicates that adapting the VLA to the target motor domain effectively establishes a language-conditioned reference policy for the brain-conditioned policy. (2) Neural-language alignment. We disable the neural-language alignment objective during training and optimize the neural encoder using only the action prediction loss. The inference pipeline remains unchanged. The performance drop shows that the alignment provides a semantic target for learning intent-related neural representations, enhancing the neural representation to guide the policy. (3) Visual observations. We remove informative vision by replacing the ground-truth rendered images with a constant white image during training and similarly replacing the evolving MuJoCo observations with the same white image during inference. This ablation therefore removes both visual state information and visual evolving feedback while leaving the remaining BrainVLA architecture unchanged. Its reduced performance highlights that visual observations supply complementary current task state, which supports the VLA policy to predict actions. (4) Adapted VLA policy. We remove the whole adapted VLA policy and replace it with a lightweight MLP decoder, while retaining its input modalities and alignment supervision. This variant excludes the adapted VLA policy as a whole, providing a direct test of large-scale VLA policy can improve action prediction. Our contribution is a framework for conditioning an adapted VLA policy on neural activity, rather than a new VLA architecture or pretraining method. The ablation experiments evaluate the effectiveness of this integration, and do not claim to isolate the effects of pretrained initialization from architectural capacity; our conclusions concern the adapted VLA policy as a whole.

Modality Ablation. As shown in Table 3, the full model consistently provides the strongest performance. Removing proprioception leads to a degradation on D1, indicating that current-state information remains important even when visual feedback is available. Removing vision causes a

Table 3: Modality ablation. The trained BrainVLA model is kept fixed while removing modalities at inference to measure their individual and complementary contributions.

Variant	Neural	Vision	Proprio	D1		D2	
				R^2	SR	R^2	SR
BrainVLA	✓	✓	✓	0.65	0.77	0.46	1.00
Neural+Vision	✓	✓	✗	0.35	0.39	0.45	0.99
Neural+Proprio	✓	✗	✓	0.27	0.29	-0.04	0.03
Vision+Proprio	✗	✓	✓	0.24	0.25	0.17	0.87
Neural	✓	✗	✗	0.15	0.11	-0.06	0.43
Vision	✗	✓	✗	0.11	0.19	0.20	0.75
Proprio	✗	✗	✓	-1.01	0.03	-30.34	0.00

larger loss in rollout performance, particularly on D2, highlighting the role of visual observations in tracking the evolving environment state. Similarly, removing neural activity weakens decoding performance, showing that environmental state alone is insufficient to recover the motor intention. Single-modality inference further reveals complementary roles across inputs. Neural activity alone provides intention-related information but lacks sufficient state feedback, while vision alone is insufficient to recover the full model's decoding performance. Proprioception alone performs poorly, suggesting that instantaneous state information is insufficient without either neural intention or visual context. Overall, these results indicate that BrainVLA benefits from the complementary combination of *neural intention*, *visual state*, and *proprioceptive state*.

4.6 Dimension-level analysis

We further analyze BrainVLA on D1 at the level of individual action dimensions, as shown in Fig. 4 (B). The full model achieves consistently positive R^2 across all seven action dimensions, indicating that BrainVLA can recover both coarse spatial motion and more fine-grained hand configurations. The modality ablations reveal a clear complementary and functional distinction between vision and neural activity. Removing vision causes the largest degradation in the translational dimensions (x, y, z), which is consistent with visual observations providing direct information about spatial configuration. In contrast, removing neural activity has a larger effect on roll and finger-related dimensions. These variables are less directly observable from the scene and depend more strongly on the intended motor command encoded in neural activity.

5 Conclusion

We presented BrainVLA, a framework that connects neural activity to an adapted VLA policy through neural–language alignment. By constructing VLA-compatible datasets and establishing a neural interface to the policy, this work provides an approach to using existing VLA models for neural motor decoding. The task-level success criteria are introduced to assess whether decoded actions accomplish the intended tasks to complement the decoding metric R^2. Across two datasets, causal rollout evaluation demonstrated strong decoding accuracy and task success on held-out sessions, alongside effective performance with limited training data. Component ablations supported the contributions of task-specific adaptation, alignment supervision, and the effectiveness of the whole adapted policy. The utilization of the VLA policy greatly improved the decoding performance, providing a promising strategy to extend neural motor decoding to a developed VLA model. Neural–language alignment not only enables the neural activity to guide the VLA policy through its language interface, but also prompts the neural encoder to extract the neural representation with latent motor intention. Modality ablations further support the complementary contributions of neural activity, vision, and proprioception to decoding performance. Dimension-level analyses further revealed complementary roles: removing vision primarily impaired translational control, whereas removing neural activity strongly affected wrist rotation and finger control. These findings demonstrate how BrainVLA realizes neural motor decoding. Finally, BrainVLA requires validation in real-time closed-loop BCI settings, where sensory feedback may alter neural activity. Our results represent an initial step toward practical neural control using adapted VLA policies.

REFERENCES

Mehdi Azabou, Vinam Arora, Venkataramana Ganesh, Ximeng Mao, Santosh Nachimuthu, Michael Mendelson, Blake Richards, Matthew Perich, Guillaume Lajoie, and Eva Dyer. A unified, scalable framework for neural population decoding. *Advances in Neural Information Processing Systems*, 36:44937–44956, 2023.

Mehdi Azabou, Krystal Pan, Vinam Arora, Ian Knight, Eva Dyer, and Blake A Richards. Multi-session, multi-task neural decoding from distinct cell-types and brain regions. In *International Conference on Learning Representations*, volume 2025, pp. 59654–59677, 2025.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control, 2026. URL https://arxiv.org/abs/2410.24164.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

Jennifer L Collinger, Brian Wodlinger, John E Downey, Wei Wang, Elizabeth C Tyler-Kabara, Douglas J Weber, Angus JC McMorland, Meel Velliste, Michael L Boninger, and Andrew B Schwartz. High-performance neuroprosthetic control by an individual with tetraplegia. *The Lancet*, 381 (9866):557–564, 2013.

Yiqun Duan, Charles Zhou, Zhen Wang, Yu-Kai Wang, and Chin teng Lin. Dewave: Discrete encoding of EEG waves for EEG to text translation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=WaLI8slhLw.

Sharlene N Flesher, John E Downey, Jeffrey M Weiss, Christopher L Hughes, Angelica J Herrera, Elizabeth C Tyler-Kabara, Michael L Boninger, Jennifer L Collinger, and Robert A Gaunt. A brain-computer interface that evokes tactile sensations improves robotic arm control. *Science*, 372(6544):831–836, 2021.

Changjiang Gao, Zhengwu Ma, Jiajun Chen, Ping Li, Shujian Huang, and Jixing Li. Increasing alignment of large language models with language processing in the human brain. *Nature computational science*, 5(11):1080–1090, 2025.

Joshua I Glaser, Ari S Benjamin, Raeed H Chowdhury, Matthew G Perich, Lee E Miller, and Konrad P Kording. Machine learning for neural decoding. *eneuro*, 7(4):ENEURO–0506, 2020.

Ariel Goldstein, Avigail Grinstein-Dabush, Mariano Schain, Haocheng Wang, Zhuoqiao Hong, Bobbi Aubrey, Samuel A Nastase, Zaid Zada, Eric Ham, Amir Feder, et al. Alignment of brain embeddings and artificial contextual embeddings in natural language points to common geometric patterns. *Nature communications*, 15(1):2768, 2024.

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, 2021. URL https://arxiv.org/abs/2106.09685.

Hongru Jiang, Zhe Liu, Xiangdong Bu, Antong Liu, Xingyu Ji, and Yao Chen. Preserved embedding of neural population activity for brain computer interfaces. *bioRxiv*, pp. 2026–02, 2026.

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.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning, 2021. URL https://arxiv.org/abs/2004.11362.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model, 2024. *URL https://arxiv. org/abs/2406.09246*, 1(2):4, 2024.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025.

Trung Le, Hao Fang, Jingyuan Li, Tung Nguyen, Lu Mi, Amy L Orsborn, Uygar Sümbül, and Eli Shlizerman. Spint: Spatial permutation-invariant neural transformer for consistent intracortical motor decoding. *Advances in Neural Information Processing Systems*, 38:25815–25835, 2026.

Johannes Y Lee, Sangjoon Lee, Abhishek Mishra, Xu Yan, Brandon McMahan, Brent Gaisford, Charles Kobashigawa, Mike Qu, Chang Xie, and Jonathan C Kao. Brain–computer interface control with artificial intelligence copilots. *Nature machine intelligence*, 7(9):1510–1523, 2025.

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.

Gavin Mischler, Yinghao Aaron Li, Stephan Bickel, Ashesh D Mehta, and Nima Mesgarani. Contextual feature extraction hierarchies converge in large language models and the brain. *Nature Machine Intelligence*, 6(12):1467–1477, 2024.

Samuel R Nason, Matthew J Mender, Alex K Vaskov, Matthew S Willsey, Nishant Ganesh Kumar, Theodore A Kung, Parag G Patil, and Cynthia A Chestek. Real-time linear prediction of simultaneous and independent movements of two finger groups using an intracortical brain-machine interface. *Neuron*, 109(19):3164–3177, 2021.

Sean M Perkins, Elom A Amematsro, John Cunningham, Qi Wang, and Mark M Churchland. An emerging view of neural geometry in motor cortex supports high-performance decoding. *Elife*, 12:RP89421, 2025.

Joséphine Raugel, Jérémy Rapin, Stéphane d'Ascoli, Valentin Wyart, and Jean-Rémi King. Scaling and context steer llms along the same computational path as the human brain. *Advances in Neural Information Processing Systems*, 38:53588–53616, 2026.

Steffen Schneider, Jin Hwa Lee, and Mackenzie Weygandt Mathis. Learnable latent embeddings for joint behavioural and neural analysis. *Nature*, 617(7960):360–368, 2023.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy, 2024. URL `https://arxiv.org/abs/2405.12213`.

Matthew S Willsey, Nishal P Shah, Donald T Avansino, Nick V Hahn, Ryan M Jamiolkowski, Foram B Kamdar, Leigh R Hochberg, Francis R Willett, and Jaimie M Henderson. A high-performance brain–computer interface for finger decoding and quadcopter game control in an individual with paralysis. *Nature Medicine*, 31(1):96–104, 2025.

Brian Wodlinger, JE Downey, Elizabeth C Tyler-Kabara, Andrew B Schwartz, ML Boninger, and Jennifer L Collinger. Ten-dimensional anthropomorphic arm control in a human brain- machine interface: difficulties, solutions, and limitations. *Journal of neural engineering*, 12(1):016011, 2015.

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

Joel Ye, Fabio Rizzoglio, Xuan Ma, Adam Smoulder, Hongwei Mao, Gary Blumenthal, William Hockeimer, Nicolas Kunigk, Dalton Moore, Patrick Marino, et al. A generalist intracortical motor decoder. *Advances in Neural Information Processing Systems*, 38:110547–110587, 2026.

Yizi Zhang, Yanchen Wang, Mehdi Azabou, Alexandre Andre, Zixuan Wang, Hanrui Lyu, International Brain Laboratory, Eva L Dyer, Liam Paninski, and Cole Lincoln Hurwitz. Neural encoding and decoding at scale. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=vOdz3zhSCj.

Yizi Zhang, Linyang He, Chaofei Fan, Tingkai Liu, Han Yu, Trung Le, Jingyuan Li, Scott Linderman, Lea Duncker, Francis Willett, et al. A cross-species neural foundation model for end-to-end speech decoding. In *International Conference on Learning Representations*, volume 2026, pp. 30773–30797, 2026.

AI USE STATEMENT

In this work, we used generative AI tools for polishing the writing of this paper and searching related works. We have reviewed all AI-assisted work. No scientific contribution or results were generated by generative AI tools. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Supplemental materials will be provided that will help with reproducibility, including constructed datasets and code of the framework.

A Dataset and Evaluation Details

A.1 Training, Validation and Test splits

We construct the data split at the session level. For each dataset, the sessions listed in Table 4 are selected as the validation set because their trials account for approximately 20% of the held-in data, allowing the remaining sessions to retain roughly 80% of the data for model training. The official held-out sessions are reserved for final testing and are not used for either model training or model selection. This protocol ensures that hyperparameter tuning and checkpoint selection are performed only on held-in data, while evaluation on the held-out sessions provides an independent assessment of generalization to unseen sessions.

Table 4: Validation session for both datasets.

D1
held-in-calib_ses-19250108T111022 held-in-calib_ses-19250119T114045 held-in-calib_ses-19250120T115537
D2
held-in-calib_ses-2020-10-27-Run2_behavior+ecephys

A.2 Task definitions and language instructions

We summarize the language instructions associated with the D1 and D2 datasets in Tables 5 and 6, respectively. D1 spans seven task labels and 18 distinct instruction strings. D2 spans initialization, outward movement, and return movement, with 17 distinct suggested instruction strings. The tables preserve the source wording while using placeholders to compactly represent repeated phrases. These placeholders are notation for this appendix and are not part of the instruction strings.

D1: object manipulation. The D1 instructions describe reaching toward a red item, orienting the wrist, shaping the hand, acquiring the item, carrying it toward a green target, reorienting it, and releasing it. The descriptions distinguish wrist rotation direction and acquisition mode (pinching, grasping, or scooping).

D2: finger-group movements. The D2 descriptions specify movements of the index finger, the MRS finger group, or both groups. The index finger is associated with a red target marker and the MRS group with a yellow target marker. Outward instructions describe movement toward the target markers, whereas return instructions explicitly describe movement back toward the center target markers. Single-group instructions also specify that the other group remains centered. For

Table 5: D1 language instructions. `[direction]` is `clockwise` or `counter-clockwise`; `[action]` is `pinch`, `grasp`, or `scoop`; and `[holding]` is `pinching`, `grasping`, or `scooping`. All six holding-direction combinations occur under `Orient2`."

Task label	Instruction template
Reach	reach toward the red item
Orient	rotate the wrist `[direction]` to align the hand with the red item
Shape	move the thumb inward spread the thumb
Grasp	`[action]` the red item
Carry	carry the red item toward the green target while `[holding]` it
Orient2	keep `[holding]` the red item and rotate the wrist `[direction]` to align with the green target
Release	release the red item on the green target

Table 6: D2 suggested language instructions. `[motion]` is `flex` or `extend`. For opposite-direction movements, `[opposite]` is `extend` when `[motion]` is `flex`, and vice versa. Each movement template represents two observed strings.

Phase	Group / pattern	Instruction template
Initialization	Both	hold both finger groups at their corresponding colored center target markers
Outward	Index	`[motion]` the index finger toward the red target marker while keeping the MRS finger group centered
Outward	MRS	`[motion]` the MRS finger group toward the yellow target marker while keeping the index finger centered
Outward	Both / same	`[motion]` both finger groups toward their corresponding colored target markers
Outward	Both / opposite	`[motion]` the index finger toward the red target marker and `[opposite]` the MRS finger group toward the yellow target marker
Return	Index	`[motion]` the index finger back toward the red center target marker while keeping the MRS finger group centered
Return	MRS	`[motion]` the MRS finger group back toward the yellow center target marker while keeping the index finger centered
Return	Both / same	`[motion]` both finger groups back toward their corresponding colored center target markers
Return	Both / opposite	`[motion]` the index finger back toward the red center target marker and `[opposite]` the MRS finger group back toward the yellow center target marker

movements involving both groups, the descriptions distinguish motion in the same direction from simultaneous flexion and extension in opposite directions.

A.3 MuJoCo environment details

A.3.1 MuJoCo Environment Details for D1 dataset

We reconstruct the motor tasks in MuJoCo using a lightweight hand workspace environment Wodlinger et al. (2015). The simulator uses a timestep of $0.02\,\mathrm{s}$ and zero gravity. The workspace contains a planar floor and three surrounding walls, together with a task board defining source and target locations for object manipulation. The hand model is represented by a movable root with three translational degrees of freedom and a wrist roll joint. Hand articulation additionally includes a thumb joint and two finger-cluster joints controlling pinch and scoop motions. The corresponding joint ranges and scene geometry are summarized in Table 7.

Task objects are represented using red cylindrical source objects and green target regions. Four source-target pairs are distributed along the task board. Additional mocap bodies are used to represent the currently active pickup object and target during replay. Two fixed cameras are defined in the scene. We use the `front` camera for the main visual observations in our experiments, while the `closeup` camera is retained for visualization and debugging. Exact camera poses are listed in Table 7. Image preprocessing and policy-query settings are described separately in the implementation details.

Visual observations were obtained from a single fixed third-person frontal camera in MuJoCo. The camera was positioned at (1.06, 0.72, 1.00) in world coordinates, with an approximately (45°) vertical field of view. RGB observations were rendered at (256×256) pixels and 50 Hz, center-cropped using 90% of the image area, and resized to (256×256) before being provided to the VLA policy. The policy was queried every eight simulation steps (6.25 Hz).

A.3.2 MuJoCo Environment Details for D2 dataset

For the 2-DoF finger-control dataset, we reconstruct the task using a lightweight MuJoCo hand model with two independently controlled degrees of freedom Nason et al. (2021). The first joint

Table 7: MuJoCo scene configuration used for D1 reconstruction.

Parameter	Setting
Simulation timestep	0.02 s
Gravity	$(0,\ 0,\ 0)$
Render size	256×256
Hand root position	$(0.52,\ 0.28,\ 0.93)$
x translation range	$[-0.55,\ 0.55]$
y translation range	$[-0.35,\ 0.35]$
z translation range	$[-0.35,\ 0.35]$
Wrist roll range	$[0.35,\ 3.14]$ rad
Thumb joint range	$[-3.14,\ 3.14]$ rad
Pinch joint range	$[0,\ 1.1]$ rad
Scoop joint range	$[0,\ 1.1]$ rad
Task-board position	$(0.0,\ 0.10,\ 0.95)$
Task-board orientation	$(0,\ 0,\ -1.6)$ rad
Source-object geometry	Cylinder, radius 0.028, half-length 0.045
Target geometry	Cylinder, radius 0.045, half-length 0.006
Number of source-target locations	4
Front camera position	$(1.06,\ 0.72,\ 1.00)$
Front camera `xyaxes`	$(0.72,\ -1.10,\ 0;\ 0.05,\ 0.02,\ -0.99)$
Closeup camera position	$(0.98,\ 0.58,\ 1.08)$
Closeup camera `xyaxes`	$(0.64,\ -0.77,\ 0;\ 0.24,\ 0.16,\ -0.96)$

Table 8: MuJoCo scene configuration used for D2 reconstruction.

Parameter	Setting
Simulation timestep	0.02 s
Gravity	$(0,\ 0,\ 0)$
Render size	256×256
Controlled DoF	Index, MRS finger group
Index joint range	$[0,\ 1.35]$ rad
MRS joint range	$[0,\ 1.35]$ rad
Input representation	Normalized position increments
Replay mapping	Cumulative sum $\rightarrow$ clip $[0, 1] \rightarrow [0, 1.35]$ rad
Center configuration	$(0.675,\ 0.675)$ rad
Extended configuration	$(0,\ 0)$ rad
Flexed configuration	$(1.35,\ 1.35)$ rad
Hand position	$(0.08,\ 0,\ 0.035)$
Hand orientation	$(-0.8236,\ 0,\ -0.3491)$ rad
Index target geometry	Sphere, radius 0.018
MRS target geometry	Sphere, radius 0.020
Front camera position	$(0.05,\ -0.70,\ 0.32)$
Top camera position	$(0.02,\ 0,\ 0.72)$
Closeup camera position	$(-0.04,\ -0.53,\ 0.25)$
Camera target	Hand body
Optional velocity gain	$k_v = 30$
Velocity control range	$[-8,\ 8]$

controls the index finger, while the second controls a coordinated middle-ring-small (MRS) finger group. Both joints are modeled as hinge joints with a range of $[0, 1.35]$ rad.

The recorded behavioral signals represent normalized position increments at 20 ms intervals. During replay, these increments are cumulatively integrated, clipped to $[0, 1]$, and linearly mapped to the corresponding joint range $[0, 1.35]$ rad. The simulator uses a timestep of 0.02 s with zero gravity. The reconstructed hand contains a fixed thumb together with articulated index and MRS finger groups. Two task targets are represented by mocap-controlled spheres corresponding to the index

and MRS movements. Fingertip sites and visual markers are defined for both controlled finger groups to facilitate target placement and visualization. The scene provides three fixed cameras—`front`, `top`, and `closeup`—all targeting the hand. Their positions and the principal simulation parameters are summarized in Table 8. Visual observations were obtained from a single fixed third-person frontal camera in MuJoCo.

A.4 TASK-SUCCESS EVALUATION CRITERIA

We evaluate task success using dataset-specific criteria applied to replayed decoder predictions. D1 uses task-dependent geometric and hand-configuration criteria in MuJoCo, whereas D2 uses endpoint error after integrating predicted finger velocities. These measures quantify simulated replay or kinematic success; they do not establish closed-loop control with feedback to the neural decoder.

D1. Each task segment receives a binary success label according to Table 9. Let d_t^{item} denote the Euclidean distance between the hand-associated item reference point and the simulated item, and let d_t^{target} denote the distance between the item and its target. Distances are measured in simulation world coordinates. Subscript T denotes the final evaluated state.

The final roll error is

$$e_T^{\mathrm{roll}} = \left| \mathrm{wrap}_{[-\pi,\pi)} \left(r_T - r_T^{\mathrm{GT}} \right) \right|,$$

computed from the recorded pose roll coordinate before its mapping to the simulated joint. Thus, this criterion compares roll with the ground-truth endpoint, rather than measuring full object-orientation error. The closure signal c_T is the final scoop coordinate when the instruction contains "scoop," the pinch coordinate when it contains "pinch," and the maximum of these two coordinates otherwise (with scoop taking precedence if both occur).

Table 9: D1 success criteria using the evaluator's default thresholds. All conditions within a row must hold.

Task	Success criterion
Reach	$\min_t d_t^{\mathrm{item}} \le 0.055$
Orient	$d_T^{\mathrm{item}} \le 0.075$ and $e_T^{\mathrm{roll}} \le 0.35$
Shape	$\lvert q_T - q_T^{\mathrm{GT}} \rvert \le 0.18$ and direction agreement as defined below
Grasp	$\min_t d_t^{\mathrm{item}} \le 0.065$ and $c_T \ge 0.60$
Carry	$d_T^{\mathrm{target}} \le 0.070$ and $c_T \ge 0.60$
Orient2	$d_T^{\mathrm{target}} \le 0.070$ and $e_T^{\mathrm{roll}} \le 0.35$
Release	$d_T^{\mathrm{target}} \le 0.075$ and $c_T \le 0.35$

For Shape, q is the pinch, scoop, or thumb coordinate selected from the instruction, in that priority order. If none is specified, the coordinate with the largest absolute ground-truth change is used. Writing $\Delta q = q_T - q_0$ and $\Delta q^{\mathrm{GT}} = q_T^{\mathrm{GT}} - q_0$, direction agreement is

$$|\Delta q^{\mathrm{GT}}| < 0.05 \quad \text{or} \quad \mathrm{sgn}(\Delta q) = \mathrm{sgn}(\Delta q^{\mathrm{GT}}).$$

Reach and Grasp use the minimum item distance over the segment; their proximity condition need not hold at the endpoint. These rules use geometric and configuration proxies without an additional contact, force, or sustained-hold requirement. Unsupported task tags are excluded from the scored set. For supported tasks, unavailable required measurements produce failure; measurement summaries use finite values and take the last finite value as the final measurement.

D2. Each trial is evaluated in the two-dimensional normalized position space of the index and middle–ring–small (MRS) finger groups. Let $\mathbf{x}_0^{\mathrm{GT}}$ be the recorded initial position, $\widehat{\mathbf{v}}_t$ the predicted per-bin displacement, and $\mathbf{g}$ the target recorded in the first pose row of the trial. For a trial containing N prediction bins, positions are reconstructed as

$$\widehat{\mathbf{x}}_0 = \mathrm{clip}_{[0,1]^2} \left(\mathbf{x}_0^{\mathrm{GT}} \right), \qquad \widehat{\mathbf{x}}_{t+1} = \mathrm{clip}_{[0,1]^2} \left(\widehat{\mathbf{x}}_t + \widehat{\mathbf{v}}_t \right), \quad t = 0, \ldots, N-2.$$

Clipping is applied independently to each coordinate. Predictions are treated as per-bin displacements, with no additional time-step factor. The success label is

$$S_{\mathrm{M2}} = \mathbb{I} \left[\left\| \widehat{\mathbf{x}}_{N-1} - \mathbf{g} \right\|_2 < 0.15 \right].$$

Table 10: OpenVLA fine-tuning configuration for D1.

Parameter	Setting
Base model	OpenVLA-7B
Fine-tuning method	LoRA + action head + proprioceptive projector
Optimization steps	50,000
Effective batch size	8
Gradient accumulation steps	1
Optimizer	AdamW
Learning rate	5×10^{-4}
Learning-rate schedule	Constant over 50,000 steps
LoRA rank	32
LoRA scaling parameter α	16
LoRA dropout	0.0
Total trainable parameters	278.76M
Training objective	L1 action regression
Action decoding	Parallel
Action chunk length	8
Action dimensionality	7
Visual input	One RGB image, 224×224
Proprioceptive dimensionality	7
Action/proprioception normalization	1st–99th percentile bounds
Image augmentation	Enabled
Numerical precision	BF16

Only the final distance determines success; reaching the target earlier does not suffice. The implementation uses $N - 1$ updates for N prediction bins, so the last predicted displacement is not integrated. Trials without corresponding pose data are skipped.

Aggregation. For each dataset, the overall success rate is

$$\mathrm{SR} = \frac{1}{|\mathcal{E}|} \sum_{i \in \mathcal{E}} S_i,$$

where $\mathcal{E}$ contains the scored D1 task segments or evaluated D2 trials. This is a pooled episode-level rate, rather than an unweighted average of task-level or session-level rates.

B Implementation and Training Details

B.1 VLA Adaptation Details

We adapt the OpenVLA-7B backbone to our neural motor datasets using parameter-efficient fine-tuning. The configuration for D1 is summarized in Table 10, and the configuration for D1 is summarized in Table 11. Three components are optimized during adaptation: the LoRA adapters inserted into the pretrained VLA backbone, the action head, and the proprioceptive projector, while the remaining backbone parameters are kept frozen. LoRA is configured with rank $r = 32$, scaling parameter $\alpha = 16$, and zero dropout. The model is trained for 50,000 optimization steps using AdamW with a constant learning rate of 5×10^{-4} and an effective batch size of 8. The action head predicts an action chunk of length 8 in parallel using an ℓ_1 action-regression objective. Overall, the adaptation setting is almost the same for the two datasets except for the action dimensionality, while preserving the majority of the pretrained OpenVLA backbone. Training used BF16 precision on one NVIDIA RTX PRO 6000 Blackwell Server Edition GPU.

B.2 Neural Encoder

The encoder follows a POYO-style latent-query architecture. For an action query at time t, the neural encoder receives spikes from the causal lookback interval $[t - 1\,\mathrm{s},\, t]$. Each spike is represented by its neural-unit identity and timestamp relative to the beginning of the lookback window. Different

Table 11: OpenVLA fine-tuning configuration for D2.

Parameter	Setting
Base model	OpenVLA-7B
Fine-tuning method	LoRA + action head + proprioceptive projector
Optimization steps	50,000
Effective batch size	8
Gradient accumulation steps	1
Optimizer	AdamW
Learning rate	5×10^{-4}
Learning-rate schedule	Constant over 50,000 steps
LoRA rank	32
LoRA scaling parameter α	16
LoRA dropout	0.0
Total trainable parameters	194.79M
Training objective	L1 action regression
Action decoding	Parallel
Action chunk length	8
Action dimensionality	2
Visual input	One RGB image, 224×224
Proprioceptive dimensionality	2
Action/proprioception normalization	1st–99th percentile bounds
Image augmentation	Enabled
Numerical precision	BF16

from common POYO, we use the same unit identity for the units across sessions, thus do not need any calibration and achieve zero-shot. Unit-start and unit-end boundary tokens are added to expose the set of recorded units, including units that do not spike in the selected window. Ordinary spikes and boundary tokens are distinguished using learned token-type embeddings. For input token i, the non-temporal representation is

$$\mathbf{x}_i = \mathbf{E}_{\text{unit}}[u_i] + \mathbf{E}_{\text{type}}[c_i], \qquad \mathbf{x}_i \in \mathbb{R}^{256}, \tag{3}$$

where u_i is the neural-unit index and c_i denotes the token type. Spike timing is incorporated through rotary time embeddings in the attention layers.

The neural encoder configuration is summarized in Table 12. Then, it uses 38 learned latent queries with timestamps uniformly distributed over the one-second encoding interval. The latent queries first cross-attend to the variable-length spike-token sequence. The resulting latent representations are then processed by four self-attention layers with eight attention heads per layer. The neural representation dimension is 256, and the final projector consists of layer normalization followed by a linear transformation:

$$\mathbb{R}^{256} \longrightarrow \mathbb{R}^{4096}. \tag{4}$$

The output of the bridge for a batch of size B is therefore

$$\mathbf{Z} \in \mathbb{R}^{B \times 38 \times 4096}. \tag{5}$$

B.3 Neural–Language Alignment

The neural-language alignment configuration is given in Table 13. During alignment, the 38 neural latent embeddings are mean-pooled to obtain a single neural representation, while the language representation is obtained by masked mean pooling over valid instruction-token embeddings from the frozen OpenVLA language embedding module. Both representations are ℓ_2-normalized before computing cosine similarity. We optimize a symmetric multi-positive contrastive objective in both neural-to-language and language-to-neural directions, where samples associated with the same instruction are treated as positive pairs. Training uses a batch size of 32 and a learnable logit scale initialized as $1/0.07$ and capped at 100.

Table 12: Neural encoder configuration.

Setting	Value
Model dimension	256
Latent queries	38
Encoder latents	38 (`num_encoder_latents=0`)
Cross-attention	1 head, head dim 64, RoPE on Q/K/V
Self-attention depth	4
Self-attention heads	8, head dim 64 (inner dim 512)
Dropout	0.1
Rotary time range	$[10^{-4}, 4.0]$, rotate dim 32
LLM projector	LayerNorm + linear $256 \to 4096$
Output shape	$[B, 38, 4096]$

B.4 Fixed-Slot Neural Prompt

We condition the adapted OpenVLA policy using a fixed-length neural soft prompt. For each action query at time t, the neural bridge maps the preceding one second of spiking activity to

$$\mathbf{Z}_t = f_\theta(\mathcal{S}_{t-1:t}) \in \mathbb{R}^{38\times 4096}, \tag{6}$$

where each of the 38 output vectors has the same dimensionality as an OpenVLA language-token embedding. The policy prompt comprises a 10-token textual prefix, 38 neural embedding slots, and a 5-token textual suffix:

$$\underbrace{\texttt{<s> In: What action should the robot take to}}_{\text{10 tokens}}\ \underbrace{[\mathbf{z}_{t,1}, \ldots, \mathbf{z}_{t,38}]}_{\text{38 neural tokens}}\ \underbrace{\texttt{?\textbackslash n Out:}}_{\text{5 tokens}} \tag{7}$$

Thus, the policy prompt always contains $10 + 38 + 5 = 53$ tokens. The policy additionally receives one RGB observation and normalized proprioception. Thus, the frozen policy is conditioned jointly on three modalities:

$$\text{policy input} = \{\text{RGB image}, \text{proprioception}, \text{neural soft prompt}\}. \tag{8}$$

The neural embeddings replace only the 38 designated language-embedding positions. Visual features are computed by OpenVLA's vision pathway, while normalized proprioception is incorporated through its proprioceptive projector, detailed shown in Table 14. OpenVLA, the action head, and the proprioceptive projector remain frozen; only the neural bridge is optimized in this experiment.

Although an auxiliary instruction string is temporarily supplied to the standard OpenVLA preprocessing pipeline to construct an action-training batch, the resulting textual prompt is subsequently discarded and reconstructed using the fixed 53-token layout. This auxiliary string therefore does not provide semantic task information to the policy.

C Baseline Implementation Details

C.1 Baseline Methods

Wiener filter. The Wiener filter Glaser et al. (2020) is a linear decoder that estimates behavior from the recent history of neural population activity. Spike trains are converted into binned firing rates, optionally smoothed, and concatenated across temporal lags to form an input vector. The behavioral estimate is computed as $\hat{\mathbf{y}}_t = \mathbf{b} + \sum_{\ell=0}^{L-1} \mathbf{W}_\ell \mathbf{x}_{t-\ell}$, where $\mathbf{x}_{t-\ell}$ denotes neural activity at lag ℓ. The parameters are fitted by minimizing squared prediction error, with ridge regularization to limit overfitting. During inference, the learned mapping directly converts the current and preceding neural observations into behavioral predictions.

RNN. Recurrent neural networks model the nonlinear temporal relationship between neural activity and behavior through a recurrent hidden state Glaser et al. (2020). At each time step, the network combines the current spike-count vector with its previous state, summarizes how neural activity has

Table 13: Neural-language alignment configuration.

Parameter	**Setting**
Embedding dimension d	4096
Batch size B	32
Neural pooling	Mean over latent slots $\mathbf{n}_i = \frac{1}{M}\sum_{m=1}^{M}\mathbf{Z}_{i,m}^{N}$
Language pooling	Masked mean over valid instruction tokens $\mathbf{t}_i = \frac{\sum_\ell m_{i,\ell}\mathbf{T}_{i,\ell}}{\sum_\ell m_{i,\ell}}$
Embedding normalization	ℓ_2 normalization $\hat{\mathbf{n}}_i = \mathbf{n}_i/\|\mathbf{n}_i\|_2, \quad \hat{\mathbf{t}}_i = \mathbf{t}_i/\|\mathbf{t}_i\|_2$
Similarity function	Cosine similarity $S_{ij} = \hat{\mathbf{n}}_i^\top \hat{\mathbf{t}}_j$
Contrastive formulation	Symmetric multi-positive
Positive-pair definition	Same instruction ID $P_{ij} = 1[y_i = y_j]$
Alignment directions	Neural→Language, Language→Neural $\mathcal{L}_{\text{align}} = \frac{1}{2}(\mathcal{L}_{N\to L} + \mathcal{L}_{L\to N})$
Logit scale	Learnable, $s = \exp(\alpha)$
Logit-scale initialization	$1/0.07 \approx 14.29$
Maximum logit scale	100
Language embedding module	Frozen OpenVLA token embeddings

Table 14: Composition of the fixed neural prompt supplied to OpenVLA. The task instruction is not included in the policy input.

Segment	Positions	Length	Content and embedding source
Text prefix	1–10	10	`<s> In: What action should the robot take to`; standard frozen OpenVLA token embeddings.
Neural slots	11–48	38	One 4096-dimensional embedding per slot, produced from causal spiking activity by the neural bridge. Placeholder lookup embeddings are completely replaced.
Text suffix	49–53	5	`?\n Out:␣`; standard frozen OpenVLA token embeddings.
Policy prompt	1–53	$10 + 38 + 5 = 53$	Fixed across tasks; no task-language tokens are supplied to the policy.
Action targets	54–109	56	Eight future actions $\times$ seven action dimensions, represented using OpenVLA action tokens during training.
Stop token	110	1	End-of-sequence supervision used during action training.
Training sequence	1–110	110	53 prompt tokens, 56 action tokens, and one stop token.

evolved during the preceding second, and a readout layer converts the resulting representation into a behavioral estimate. Our implementation uses a unidirectional long short-term memory (LSTM) network, whose gates regulate the retention and updating of temporal information. The model is trained end-to-end using mean squared error between predicted and observed velocities.

CycleGAN. CycleGAN addresses cross-session recording variability by learning an unsupervised transformation between source-session and target-session neural activity. Two generators map activity in opposite directions, while two discriminators encourage the translated activity to match the corresponding session distributions. A cycle-consistency loss encourages reconstruction of the original activity after the two mappings are applied consecutively, reducing arbitrary transformations. A behavioral decoder is first trained using labeled source data and then frozen. At inference, target-session activity is mapped into the source representation and passed to the fixed decoder. This procedure requires target neural activity for alignment but does not require target behavioral labels.

NoMAD. Nonlinear Manifold Alignment with Dynamics (NoMAD) stabilizes decoding by aligning neural activity through a shared model of latent dynamics. The original method first fits a Latent Factor Analysis via Dynamical Systems (LFADS) model to reference-session recordings and trains a behavioral decoder on the inferred latent activity. For a new session, an alignment network learns to map the observed neural activity into coordinates compatible with the reference dynamics. Unsupervised objectives encourage agreement with the reference latent distribution and reconstruction of

the observed spikes, while the reference dynamics and behavioral decoder remain fixed. The aligned latent trajectories are subsequently decoded using the reference behavioral mapping.

SPINT. The Spatial Permutation-Invariant Neural Transformer (SPINT) Le et al. (2026) treats a recorded neural population as an unordered set of units. A shared neural identity encoder extracts context-dependent unit embeddings from unlabeled calibration activity. These embeddings are combined with each unit's recent activity, and cross-attention aggregates information across units to predict behavioral variables. Dynamic channel dropout during training exposes the model to different population compositions and encourages robustness to missing units. When applied to a new session, SPINT computes new unit embeddings from unlabeled neural observations while keeping its learned parameters fixed, enabling adaptation without gradient updates or behavioral labels.

NDT2. Neural Data Transformer 2 (NDT2) Ye et al. (2023) learns transferable representations of spiking activity through multi-context pretraining. Neural recordings are discretized into spike counts, and groups of channels are combined into spatial patches at each time step. The resulting tokens encode neural activity together with temporal, spatial, and recording-context information. During pretraining, a subset of tokens is masked, and the model learns to reconstruct the missing activity using a spike-count likelihood objective. The pretrained representation is subsequently adapted for downstream tasks, including behavioral decoding. This process allows information learned across recording sessions and subjects to support decoding with limited task-specific data.

NDT3. Neural Data Transformer 3 (NDT3) Ye et al. (2026) extends large-scale neural modeling through autoregressive training on paired neural activity and behavior. Binned spike counts are grouped into spatial patches, while behavioral dimensions are represented as separate tokens. Modality-specific projections and position embeddings organize these tokens into a sequence processed by a causal Transformer. The model learns to predict subsequent tokens from the available preceding context, allowing neural and behavioral information to be modeled jointly. After pretraining across diverse recordings, the model is fine-tuned on the downstream task to produce behavioral predictions. Evaluation must specify whether previous behavioral inputs are observed, predicted, or unavailable, because this choice changes the decoding setting.

POYO. POYO is a neural population decoding framework that represents individual spikes as tokens associated with neuronal identity and event time Azabou et al. (2023). An input cross-attention module compresses the variable-length spike sequence into a fixed collection of latent representations, which are processed by latent self-attention layers. Output queries associated with the desired decoding times then attend to these representations to generate behavioral predictions. The model is trained using paired neural recordings and behavioral targets across sessions. Its tokenization and latent bottleneck support recordings with different neuronal populations, while adaptation to a new recording can involve learning new unit embeddings and fine-tuning the decoder.

C.2 Reproduction and Causal Adaptation Details

The reproduced experiments for all baseline methods use the same held-in training and validation sessions, and held-out calibration recordings for evaluation as our methods. The documented implementations include a ridge-regularized Wiener filter, a two-layer LSTM with 256 hidden units per layer, CycleGAN alignment followed by a frozen Wiener filter, and SPINT with context-dependent unit embeddings. The local NoMAD baseline is a causal GRU-based adaptation with maximum mean discrepancy alignment, and the local NDT2 baseline is a compact causal spatiotemporal Transformer trained from scratch; these should therefore be identified as NoMAD-inspired and NDT2-style implementations, respectively. CycleGAN and the NoMAD-inspired model use unlabeled held-out neural activity for alignment, while SPINT uses it to infer unit identities without parameter updates; these constitute transductive calibration settings. Neural models in this experiment are trained from scratch for at most 30 epochs, with early stopping after six non-improving validation evaluations. Performance is measured using variance-weighted R^2 in original velocity units and task-specific simulated replay or endpoint success.

Table 15: VLA adaptation with language and vision (oracle policy). Task head specifies learnable action heads and proprioception projectors.

Policy	D1 (R^2)	D1 (SR)	D2 (R^2)	D2 (SR)
OpenVLA-7B + task heads	0.86±0.03	0.85±0.06	0.39±0.02	0.99±0.01
OpenVLA-7B + LoRA + task heads	0.89±0.02	0.97±0.04	0.49±0.02	1.00±0.00

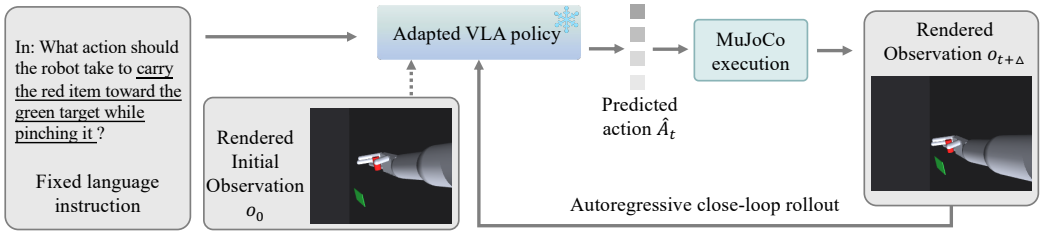

Figure 5: Language-conditioned closed-loop rollout protocol. At each step, the fixed language instruction and the current MuJoCo-rendered observation are used to predict action $\hat{A}_t$. Executing $\hat{A}_t$ updates the environment and produces the next rendered observation $o_{t+\Delta}$, which is fed back to the VLA policy. Thus, inference uses only past fixed language instruction and self-evolving visual observations generated by the model's own actions.

D Additional Quantitative Results

D.1 Language-Conditioned Closed-Loop Rollout results

We first evaluate whether the pretrained VLA backbone can be adapted to our motor tasks before applying neural conditioning. In this oracle setting, the policy receives the language instruction and the current visual observation, shown in Fig. 5. Because the native OpenVLA action does not directly match the action spaces of D1 and D2, both policies in Table 15 use a frozen OpenVLA-7B backbone with a learnable task-specific action head and proprio projectors. The second additionally introduces LoRA modules to adapt the pretrained VLA representation. The results show that task-specific adaptation produces an effective visuomotor policy for BrainVLA, while LoRA further improves its performance.

D.2 Brain-conditioned Closed-Loop Rollout results

Due to space limitations, the main paper presents only a small number of representative closed-loop rollouts. Additional qualitative results for both D1 and D2 are provided in the appendix, as shown in Figs. 6 and 7. These examples cover a broader range of reaching, grasping, object manipulation, wrist rotation, and finger-control behaviors. In each rollout, BrainVLA predicts an action from the causal neural history together with the current visual observation, and the predicted action is executed in MuJoCo to generate the next observation. The additional examples further illustrate the temporal evolution of neural-conditioned behavior under autoregressive visual feedback.

D.3 Task-Level Analysis

We further analyze BrainVLA on D1 at the level of individual task types, as shown in Fig. 8. The full model achieves consistently high success rates across the evaluated tasks, indicating that BrainVLA can recover both coarse spatial motion and more fine-grained hand configurations.

In the dimension-level analysis, the modality ablations reveal a clear functional distinction between vision and neural activity as shown in 4 (B). Removing vision causes the largest degradation in the translational dimensions (x, y, z), which is consistent with visual observations providing direct information about spatial configuration and object-target geometry. In contrast, removing neural activity has a substantially larger effect on roll and finger-related dimensions, including upper-finger,



Figure 6: Representative closed-loop BrainVLA rollouts on D1. Each row shows sequential observations from one neural-conditioned episode, progressing from left to right. At every step, BrainVLA predicts an action from the causal neural history and current rendered observation; the action is executed in MuJoCo to generate the next observation. The examples illustrate progressive reaching, object interaction, and target-directed manipulation under autoregressive visual feedback.

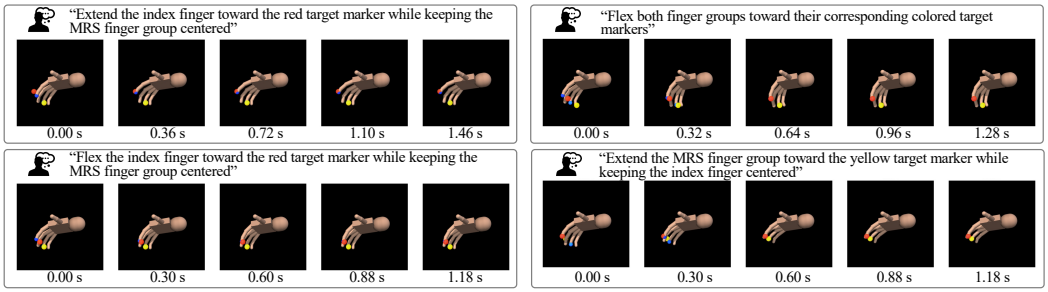

Figure 7: Representative closed-loop BrainVLA rollouts on D2. Each row shows sequential observations from one neural-conditioned episode, progressing from left to right. At every step, BrainVLA predicts an action from the causal neural history and current rendered observation; the action is executed in MuJoCo to generate the next observation. The examples illustrate progressive reaching, object interaction, and target-directed manipulation under autoregressive visual feedback.

lower-finger, and thumb control. These variables are less directly observable from the scene and depend more strongly on the intended motor command encoded in neural activity.

A similar pattern emerges at the task level. Tasks such as *Reach* and *Carry*, which depend strongly on spatial displacement, are particularly sensitive to the removal of vision. By comparison, orientation-, shape-, grasp-, and release-related tasks degrade more strongly when neural input is removed, reflecting their greater dependence on fine-grained hand configuration and internally specified motor

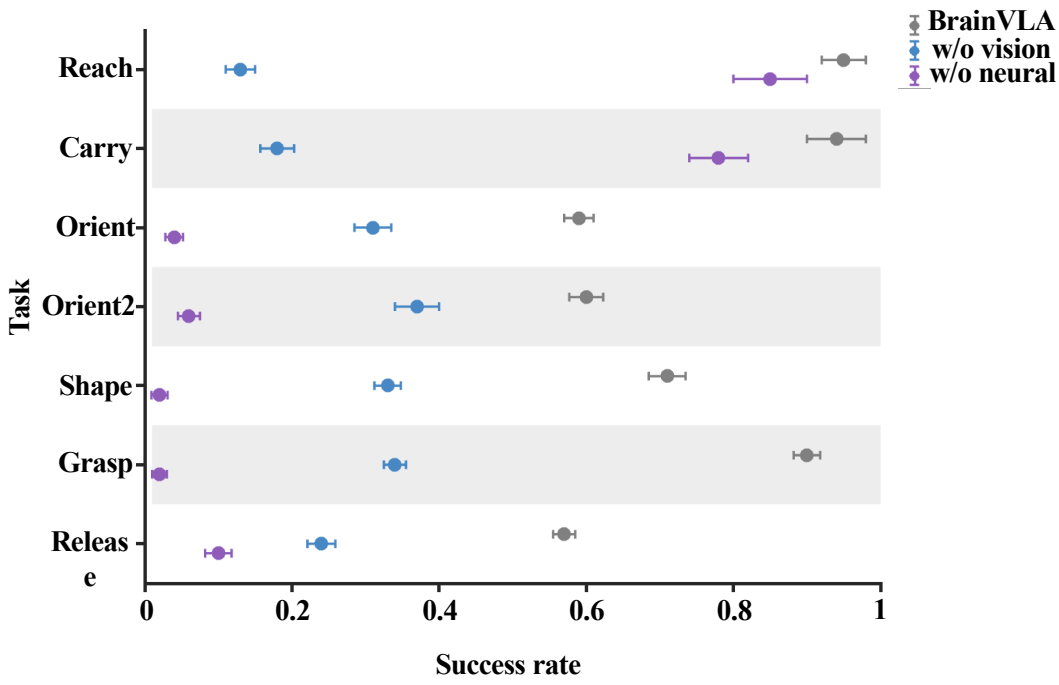

Figure 8: Task-level analysis on D1. Task success rates under the full BrainVLA model and ablations without vision or neural input. Reach and Carry are more sensitive to vision removal, while other tasks rely more strongly on neural conditioning. Error bars denote variability across test sessions.

intent. Overall, these results suggest that vision and neural activity provide complementary information: visual observations primarily support estimation of the external task state, whereas neural activity contributes intention-related information required for more subtle motor control.

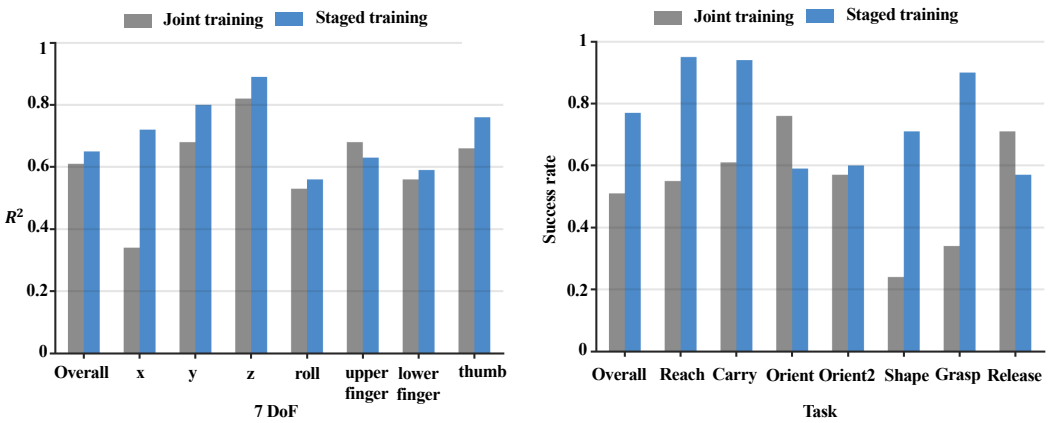

Figure 9: Comparison of joint and staged training on D1. Left: overall and dimension-wise action R^2 for the 7-DoF action space. Right: overall and task-specific success rates. Staged training provides stronger and more consistent performance overall, supporting the sequential adaptation strategy used in BrainVLA.

D.4 STAGED VS. JOINT TRAINING

We further examine whether the VLA adaptation and neural conditioning stages should be optimized jointly or sequentially. In our default *staged training* strategy, we first adapt the VLA policy using language conditioning as described in Fig. 1(B), and subsequently train the neural encoder to replace language conditioning in Fig. 1(C). In contrast, *joint training* skips the language-conditioned adaptation stage: the neural encoder, LoRA adapters, action head, and proprioceptive projector are optimized simultaneously, with neural representations directly conditioning the OpenVLA backbone throughout training.

As shown in Fig. 9, staged training yields stronger overall trajectory decoding and task success on D1, with particularly clear improvements for spatial dimensions and several manipulation tasks. Joint training remains competitive for some dimensions and tasks, but exhibits less consistent performance across the full action space. These results support our choice to first establish a task-adapted VLA policy and then learn the neural-to-policy interface, rather than adapting both simultaneously from neural supervision.

One possible explanation is that language provides a more stable semantic conditioning signal during VLA adaptation, allowing the policy to first acquire the target visuomotor behavior before introducing the noisier and more heterogeneous neural representation. The subsequent neural-language alignment can then map neural activity into an already established conditioning space, reducing the burden of jointly learning task adaptation and neural conditioning.

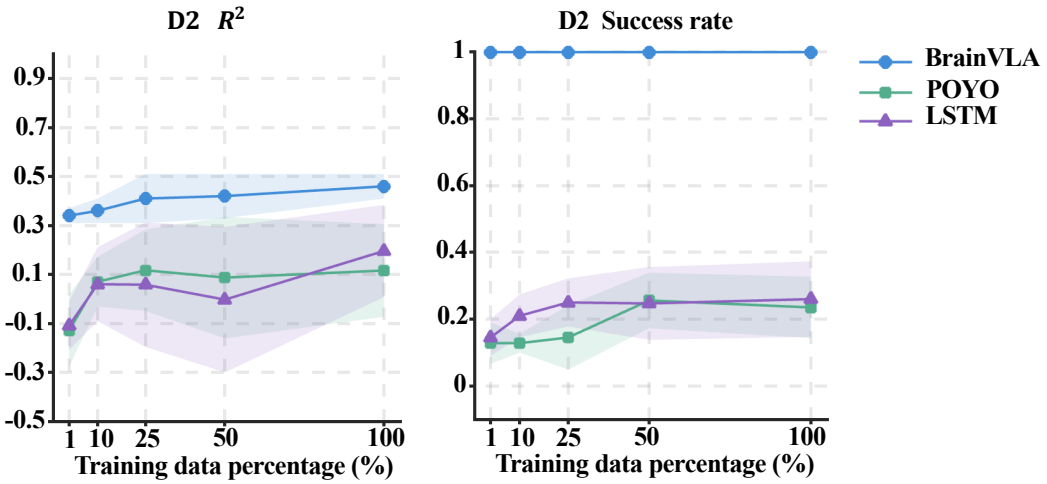

Figure 10: R^2 and task success rate with scaled training D2 data. BrainVLA consistently outperforms POYO and LSTM across data regimes and remains effective under limited neural supervision. Shaded regions indicate variability across test sessions.

D.5 DATA-EFFICIENT VALIDATION ON D2

We also evaluate data efficiency by training each model with diverse fractions of the D2 training set. As shown in Fig. 10, BrainVLA consistently outperforms POYO and LSTM across data regimes in both trajectory R^2 and task success rate. BrainVLA improves steadily with increasing training data, while the baselines exhibit substantially lower performance and greater variability, particularly in the low-data regime. Notably, BrainVLA retains meaningful decoding and task-level success even when trained with only a small subset of data. These results suggest that our policy improves the data efficiency of neural motor decoding.

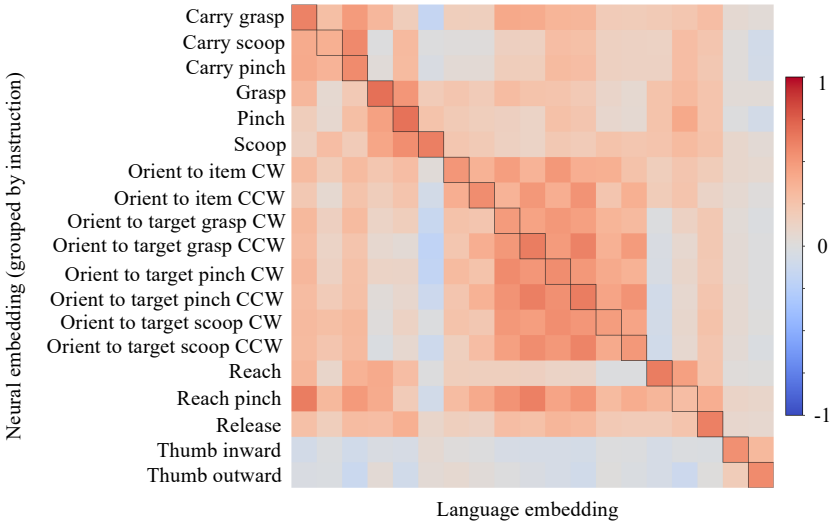

Figure 11: Alignment between neural representation and language representation. Warmer colors denote higher representational similarity.

D.6 Neural–Language Alignment Visualization

Figure 11 visualizes the alignment between neural representations and the representations of their corresponding language instructions. For each instruction, the neural latent representations are mean-pooled across latent slots, while the language token representations are mean-pooled over valid instruction tokens. Both pooled representations are ℓ_2-normalized, and each entry in the similarity matrix is computed using their cosine similarity. Rows correspond to neural representations grouped by instruction, and columns correspond to language representations. The outlined diagonal entries indicate the matched neural-language pairs, while warmer colors denote higher representational similarity.