Retrieval-Augmented Inverse Dynamics for Robotic Manipulation

Sammy Heinz, Constantin Ertel, Nicholas Fitzpatrick, JP Schuchter

INDENG 242B · May 2026 · Department of Industrial Engineering and Operations Research
University of California, Berkeley

Abstract

Generalist robot policies have converged on two competing paradigms. Vision–language–action (VLA) models map images and language instructions directly to motor commands with strong semantic generalization. In parallel, a growing research consensus argues that robots must first internalize physical world dynamics before acting on them. World models internalize this cause-and-effect through latent state predictions, but inferring the motor commands from imagined ("dreamed") future visual states is a core issue. This report studies the inverse-dynamics step: given the current visual state and a predicted next visual state, what action should the robot take? We use GR-1 as a synthetic world model to imagine the next state. We implement ***Retrieval-Augmented Inverse Dynamics (RAID)****, a head that maps the (current, dreamed) latent pair to continuous 7-DOF actions by combining a parametric trunk with a cross-attention prior over k = 3 retrieved demonstrator actions. On LIBERO-Spatial, RAID achieves a normalized validation MSE of 0.131 vs. 0.852 for the direct visual MLP with only 25 demonstrations, a 6.5x improvement. A GRPO probe reaches the best mean reward of 1.226 and a peak of 25% success, showing that retrieval-based inverse dynamics can be refined through interaction with the simulator, but more training is required. These results suggest that RAID might be a viable lightweight alternative to fully learned action decoders for world model command inference, potentially enabling better one-shot learning.*

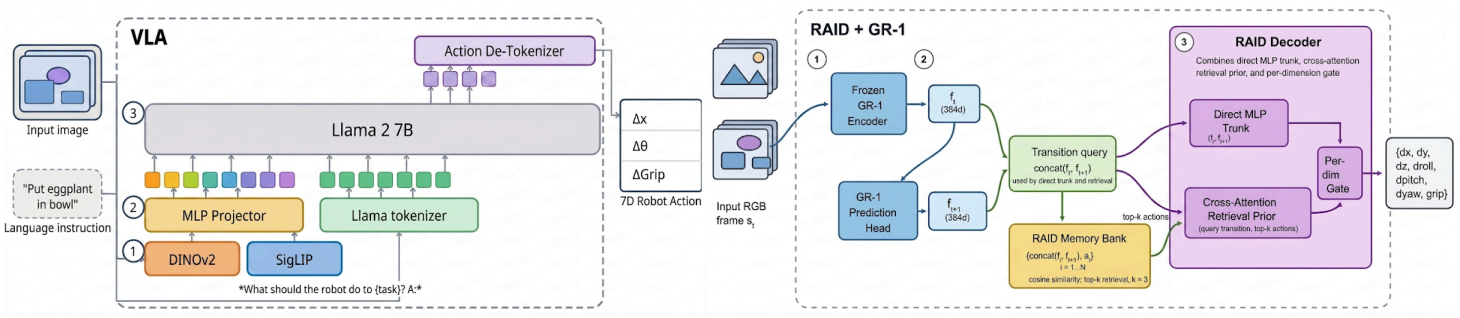


Figure 2. Left: VLA (Llama-2-style) Right: RAID + GR-1

1. Motivation and Related Work

State of AI Robotics & VLAs Humanoids are increasingly gaining traction, e.g., performing baggage handling at Japanese airports and managing high-volume logistics for UPS and BMW. Two paradigms currently dominate generalist robot policies. Vision-language-action (VLA) models (Brohan et al., 2023), use large-scale pre-training to map multimodal instructions to actions. However, VLAs lack explicit causal grounding, as they fail to internalize physical dynamics required for robust spatiotemporal reasoning.

World Models In parallel, world-model approaches argue that robots must internalize physics before acting. V-JEPA (Bardes et al., 2024) provided a technical foundation for this approach by predicting latent representations of masked spatiotemporal tubelets. The premise is that a model that can "dream" the next state, meaning predicting abstract latent features rather than raw pixels to master physical

cause-and-effect, has acquired an inductive bias for how the world evolves. World models have recently gained substantial popularity in robotics, e.g. NVIDIA (Cosmos, NVIDIA 2025; DreamZero, 2026) and Google DeepMind (Genie, Bruce et al., 2024; DreamerV3, Hafner et al., 2023).

Command Inference Dreaming a next state does not produce the motor command to reach it. Inferring these commands is particularly difficult for manipulation with seven degrees of freedom (7-DOF), corresponding to three translation axes, three rotation axes, and one gripper opening dimension, where visually similar next states diverge wildly under compounding contact dynamics, forcing networks to generalize from limited data. As Professor Ken Goldberg framed it in personal correspondence: "world models and JEPA are interesting, and many are using them for robot locomotion, but their success has been limited for robot manipulation as it's very hard to infer the associated robot commands."

EFT To address this, we draw inspiration from episodic future thinking (Atance & O'Neill, 2001). When humans face a decision, the hippocampus imagines plausible future states by recombining past scenes, and episodic memory constructs the actions that navigated those scenes from retrieved experience. Hassabis made this architecture central to his vision of AI, arguing that experience replay in DQN was a first crude instantiation of it, and that episodic control over stored transitions is a more faithful approximation of how biological agents act (Hassabis et al., Neuron 2017).

RAID RAID instantiates this mechanism for robotic manipulation. Given a world model that imagines future states, RAID uses cross-attention (Vaswani et al., 2017) to retrieve the most structurally similar transitions from a demonstration memory bank and decodes the causing action. Retrieval-augmented generation has been explored in embodied AI more broadly (Zhu et al., 2024), but to our knowledge, this is the first direct application of retrieval-augmented generation (Lewis et al., 2020) to the inverse-dynamics sub-problem of mapping (s_t, s_{t+1}) to a_t.

Current Solutions Contemporary robotics world models, e.g., DreamGen (Jang et al., 2025) and DreamZero (Ye et al., 2026), identified the action inference issue. They rely on fully learned network weights to infer actions. RAID differs by grounding inverse dynamics in retrieved demonstrations, offering a lightweight alternative that enables rapid task adaptation without extensive retraining. Finally, to bridge the gap from imitating recorded demonstrations (offline training) to actual task completion, we use Group Relative Policy Optimization (GRPO) to fine-tune the retrieval-augmented policy in active simulation (Shao et al., 2024). We selected GRPO because its group-relative scoring circumvents the need for the parameter-heavy value networks.

2. Methodology

2.1 Problem statement and notation

Let f_t denote a GR-1 vision-feature observation and a_t the action applied at step t. Given a memory bank $M = \{(f_i, f_{i+1}, a_i)\}$ of demonstrated transitions, the inverse-dynamics task is to predict a normalised action $\hat{a}_t$ from (f_t, f_{t+1}). At deployment, GR-1 supplies a one-step-ahead dreamed feature $\hat{f}_{t+1}$ that stands from the current observation f_t. The RAID head then maps $(f_t, \hat{f}_{t+1})$ to a 7-DOF action.

2.2 GR-1 + Cross-Attention RAID

Figure 1 captures our architecture compared to a typical VLA (left figure from OpenVLA paper, Kim et al., 2024). GR-1 is a language-conditioned vision Transformer pretrained on large-scale Ego4D-style video to predict future image features and actions (Wu et al., 2024). We turn it into a synthetic world model. We drop its language/action-output role, freeze the visual encoder, and use its 384-dimensional class-token feature as the current state f_t while its one-step prediction head provides the dreamed next state f_{t+1}. The RAID decoder takes those two GR-1 features as its input and produces the predicted action. We add a trunk that ensures stability when the memory bank lacks similar neighbors, and a gate that dynamically prioritizes the retrieved prior for joints like the gripper that require high-precision templates:

(i) Parametric $d_\varphi(f_t, f_{t+1})$ MLP trunk. Provides a direct action estimate. Two-hidden-layer MLP over concat(f_t, f_{t+1}); LayerNorm + ReLU + dropout 0.1.

(ii) Cross-attention prior. For a query transition, we retrieve the top-k most similar memory entries by cosine similarity in the joint (f_t, f_{t+1}) space and compute a pooled prior $\hat{a}_{prior}$ from the retrieved actions. This formula calculates a weighted average of retrieved demonstrator actions by comparing a query derived from current and dreamed latent features against keys from the memory bank to produce an action template. Project concat(f_t, f_{t+1}) to a query $q \in \mathbb{R}^d$ and each retrieved action a_i to a key $k_i \in \mathbb{R}^d$:

$$\alpha_i = \mathrm{softmax}\left(\frac{q^T k_i}{\sqrt{d}}\right), \quad \hat{a}_{\mathrm{prior}} = \sum_{i=1}^{n} \alpha_i \cdot a_i$$

The values are the actions themselves, so the prior remains in the 7-D action space and stays interpretable. query and keys come from different sources (state vs. retrieved actions).

(iii) Per-dimension gate. This formula uses a learned, per-dimension sigmoid gate to adaptively blend the direct action prediction from the MLP trunk with the retrieval-augmented action prior. The trunk and prior are blended dimension-wise:

$$\hat{a} = g \odot d_\varphi(f_t, f_{t+1}) + (1 - g) \odot \hat{a}_{prior}, \quad g = \sigma(W\,concat(f_t, f_{t+1}) + b)$$

At training time we apply prior dropout (p = 0.4) and additive Gaussian jitter (sigma = 0.05) to the retrieved prior in the final visual model. Without this regularization, the optimizer can copy the retrieved prior too aggressively. Dropout and noise force the direct trunk and the gate to remain useful when retrieval is imperfect.

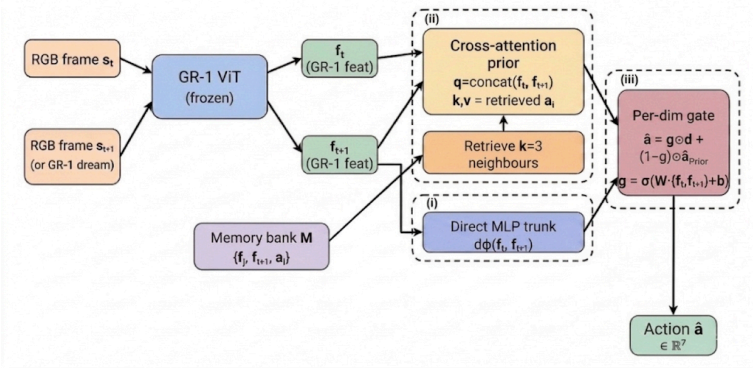


Figure 3. **Overview of the gated GR-1 and Cross-Attention RAID architecture for refined action priors.**

2.3 Refinement with GRPO

To bridge the gap between low validation error on recorded data and the policy actually solving the task when run in the simulator, we employ GRPO on the first LIBERO-Spatial pick-and-place task. We chose GRPO because it avoids training a critic or value function: each update samples a small group of rollouts (full task attempts run from the same starting state, where the policy adds random noise to its actions so each rollout explores a slightly different trajectory), scores them with the same task reward, and converts within-group relative performance into the policy-gradient advantage. Further implementation details are in Appendix D.

$$L_{GRPO}(\theta) = -\,E_i\,[\,A_i \cdot \text{mean}_t \log \pi_\theta(a_{i,t} \mid s_{i,t})\,] \qquad A_i = \frac{R_i - \text{mean}(R_1, \ldots, R_G)}{\text{std}(R_1, \ldots, R_G) + \varepsilon}\,,\ G = 4$$

Here, R_i is the total rollout reward for rollout i, G is the group size (G = 4 in our runs), and A_i is the normalized group-relative advantage. In implementation, we use the mean trajectory log-probability under the current RAID policy, an approximate KL/log-probability drift penalty to the frozen RAID reference, and log-standard-deviation regularization to prevent unstable exploration.

3. Computational Pipeline

3.1 Dataset and observations

LIBERO-Spatial (Liu et al., 2023) provides 10 pick-and-place tasks involving 9 objects across different layouts, with 50 demonstrations per task and a synchronised RGB camera plus 7-DOF action (dx, dy, dz, $d\theta_x$, $d\theta_y$, $d\theta_z$, grip). We obtained the demonstrations from the LIBERO HDF5 release, verified that all 10 task files contained the expected 50 demonstrations, skipped terminal frames without valid next observations, and built a cached transition dataset from consecutive frame pairs. Each cached record stores f_t, f_{t+1}, action, task id, demo id, and step id, which lets training reuse GR-1 features without repeatedly running the encoder. Actions are normalized per dimension using a mean and standard deviation computed once from the training split. The same normalization statistics are reused at validation and evaluation time. Splits are by demonstration (80 / 20) so trajectories never leak across the training and validation set. The final feature cache contains roughly 62k transitions and occupies about 180 MB. The complete compute and infrastructure budget can be found in Appendix F.

3.2 Training objective and recipe

We optimize the mean-squared error on normalized actions plus an L_2 weight penalty. Let ψ represent all learnable parameters of the RAID head (including the direct trunk φ, attention projections, and gate weights). The training objective over N demonstration transitions is:

$$\min_{\psi} \frac{1}{N} \sum_{i=1}^{N} \left\| d_\psi \left(f_t^{(i)}, f_{t+1}^{(i)}, \mathcal{M} \right) - a_t^{(i)} \right\|^2 + \lambda \|\psi\|^2$$

AdamW (β_1 = 0.9, β_2 = 0.999, lr 1 × 10^{-3}, weight decay 10^{-4}), batch size 256, 100 epochs, best-validation-MSE checkpointing. The memory bank is implemented as dense PyTorch tensors: concatenated (f_t, f_{t+1}) keys are L2-normalized, queried by cosine similarity via matrix multiplication, and retrieved with torch.topk.

4. Results & Discussion

4.1 Headline result: GR-1 + cross-attention RAID on LIBERO-Spatial

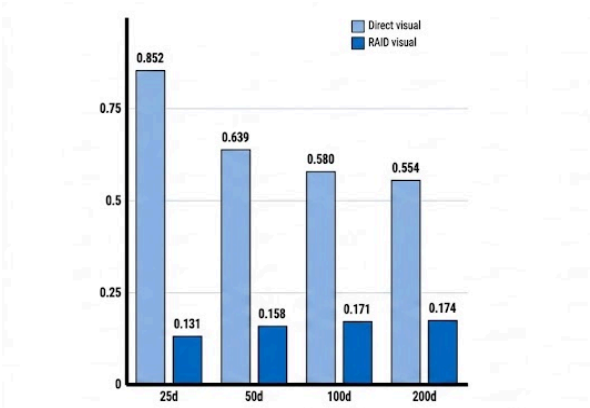


Figure 5. RAID vs. Direct Visual MSE across demos

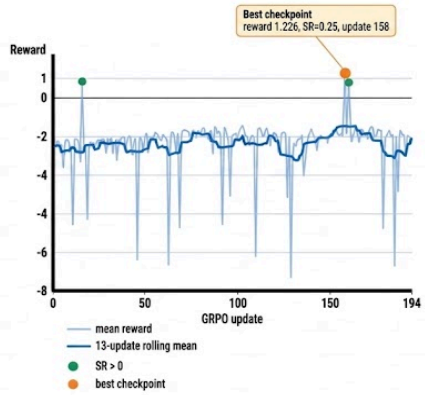


Figure 6. GRPO Rollout Reward

At N = 25 demonstrations on LIBERO-Spatial, cross-attention RAID over GR-1 features attains **0.131 validation MSE** against **0.852** for the same visual decode without retrieval, a **≈ 6.5× improvement**. A retrieval benefit persists through N = 200, although the gap narrows as the direct model receives more data. Training loss curves can be found in Appendix B.6.

4.2 Early Experiments

Before settling on GR-1, we ran a 53-cell, 171-run matrix on RoboMimic and pooled LIBERO using DINOv2 (Oquab et al., 2023) and SigLIP (Zhai et al., 2023) features under direct MLP, mean-pool RAID, cross-attention RAID, a 4-frame causal Transformer IDM, and a Diffusion-Policy IDM (Ho et al., 2020; Chi et al., 2023). Full results and setup can be found in Appendix E.

Three findings from those runs motivated the switch to GR-1. First, on the simplest setting (RoboMimic Lift, low-dim, N = 25–200), the direct MLP achieved lower error than RAID at every data scale (Table 1). Second, on RoboMimic Square with DINOv2 features, cross-attention RAID is statistically tied with the direct MLP at the native 20 Hz sampling rate (consecutive frame pairs) but becomes the lowest-MSE head once we keep only every fifth frame (temporal stride 5), because the longer time gap produces a more visible change between consecutive frames. This is a crucial qualitative shift that GR-1 amplifies. Third, on pooled LIBERO-Spatial with a 10-task DINOv2 memory bank, retrieval does not reach the error of the direct MLP, ruling out "more memory" as the explanation for the GR-1 improvement. These results pushed us to focus on changing the encoder rather than expanding the memory bank.

Condition	N=25	N=50	N=100	N=200

kNN (k=3, no training)	0.617	0.744	0.717	0.567
Direct MLP	0.341 ± 0.011	0.357 ± 0.004	0.298 ± 0.003	0.179 ± 0.004
RAID (gated + dropout + noise)	0.403 ± 0.005	0.392 ± 0.004	0.341 ± 0.004	0.215 ± 0.004

Table 1. RoboMimic Lift, low-dim, validation MSE on normalised actions Mean ± SD across three seeds.

4.3 Takeaways

The 6× MSE reduction is the clearest evidence we have that RAID fundamentally adds to addressing the action inference gap. GR-1's pretraining objective, predict future image features, is exactly what produces the temporal-delta signal the inverse-dynamics head needs. Without that signal (early experiments), retrieval has no traction even with a larger bank. With it, the cross-attention prior places most of its weight on a single coherent neighbour and supplies a cleaner action template than the parametric trunk could learn on its own from only 25 trajectories. The gap between RAID and direct MLP shrinks as the number of demos increases. This suggests RAID is primarily a sample-efficiency mechanism, not a guaranteed asymptotic improvement. Retrieval helps most when the model can't learn reliable action mappings from few demonstrations, but as coverage improves, the direct baseline benefits more from scale and RAID may inherit bias from imperfect nearest-neighbor matches. Appendix C provides a visual check: across 16 randomly sampled test transitions, RAID's predicted action matches the sign and relative magnitude of the ground-truth dominant dimension more consistently than the direct trunk. Video demonstrations can be found here: https://constantinvictorbeatertel.github.io/RAID/

4.4 GPRO

The GRPO run improved closed-loop shaped reward but did not yet produce a stable solved policy. Starting from the N = 200 behavior-cloned RAID checkpoint, the GRPO fine-tuning run logged 195 updates, reached best mean reward 1.226 at update 158, and on a subset of those updates reached 25% success, meaning that one out of the four rollouts in that group completed the task. Across the run as a whole, that success rate is evidence that the RAID prior is useful for further training inside the simulator (online refinement), but sparse manipulation success remains unreliable across rollouts. This is a reminder that low imitation error on recorded data does not by itself translate into the robot completing tasks, and that retrieval works best when it is paired with a closed-loop reward signal that can prune the cases where the retrieved action looks reasonable but does not actually work.

4.5 Broader Implications

RAID is a lightweight, non-parametric way to handle the action-inference step in robotic manipulation. The memory bank gives it three practical properties worth testing at a larger scale. Top-k retrieval is cheap at inference time, swapping demonstrations into the bank gives the model a path to one-shot task transfer, and adding a new task does not require overwriting model weights to the same degree as fine-tuning a fully parametric policy would. The broader implication, if these properties hold up at scale, is that the visual imagination part of a policy can be decoupled from the task-specific action priors. Robots could then adapt to new tasks by swapping or expanding memory banks instead of retraining large end-to-end policies from scratch.

5. Discussion of Safety, Security, and Ethics

Physical Safety A RAID decoder is still a learned policy, and it will issue a motor command whether or not the dreamed next state was accurate, whether or not the retrieved demonstrations matched the current scene, and whether or not the current observation looks like anything it saw in training. In a simulation the cost of getting one of these wrong is abstract. On a physical robot, the same wrong command can cause damage to items or humans. Before this class of model gets implemented in the real world, it needs a series of fail-safes, controls, and human oversight to account for that.

Misalignment The same architecture that lets a robot pick up a bowl lets it carry out any sufficiently well-specified instruction, and a robot that follows language instructions without a reliable way to refuse one is only as safe as the process generating those instructions. If that process is another model whose behavior is not well-constrained, the worst case is bounded by what the robot can physically do rather than by what anyone intended. One advantage of RAID is that the retrieval bank is an inspectable component, so unsafe demonstrations can be removed from it and retrieval can be filtered.

Replacement & Economic Inequality Easing the barriers to robotic automation lowers the relative cost of replacing physical labor, and the workers most exposed to this are often low paid (Acemoglu & Restrepo, 2020). The gains from AI, meanwhile, accrue disproportionately to the shareholders of the firms that deploy it, and the AI-enablers (e.g. model providers, data centers, chips) higher up the stack. Potential solutions, e.g. redistribution, retraining, are outside the scope of this paper.

6. Limitations and Future Work

Our headline metric is offline validation MSE on normalized actions, but contact-rich manipulation still requires closed-loop reward feedback, exploration control, and more stable policy updates before it can be solved reliably. In addition, one could use a world model specifically trained for our framework to get a better indication of the true usefulness of RAID. We have also not attempted to scale GR-1 + RAID to multi-task or cross-embodiment deployment in the way DreamGen and DreamZero have. Further, future work should make the retrieval gate more adaptive, tune or learn k, and down-weight retrieval when memory-bank similarity is low to address the shrinking gap between RAID and direct MLP (or the increasing MSE for RAID) as the number of demos increases.

7. Conclusion

The action-inference gap is real and the core problem this report isolates. RAID is one modular way to solve it. GR-1 produces a representation of the next state, the memory bank stores action priors from demonstrations, and the learned trunk and gate define how much to rely on the retrieved action versus the parametric estimate. Empirically, this structure reduces the 25-demo LIBERO-Spatial validation MSE from 0.852 to 0.131 compared to direct MLP and remains better than the baseline through 200 demonstrations. This indicates that RAID has true application potential as a lightweight alternative to learned networks. The GRPO fine-tuning result, however, shows that while behavior-cloned policy can be improved, the policy still only succeeds on a fraction of rollouts.

Appendix

Presentation link: https://youtu.be/Q46po2-EAq8

References

Acemoglu, D., & Restrepo, P. (2020). Robots and Employment: Evidence from US Labor Markets. *Journal of Political Economy*, 128(6), 2188–2244

Bardes, A., Garrido, Q., Ponce, J., Chen, X., Rabbat, M., LeCun, Y., Assran, M., & Ballas, N. (2024). Revisiting Feature Prediction for Self-Supervised Visual Representations from Video (V-JEPA). Meta FAIR.

Brohan, A., Brown, N., Carbajal, J., Chebotar, Y., et al. (2023). RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control. CoRL.

Bruce, J., Dennis, M., Edwards, A., Parker-Holder, J., Shi, Y., et al. (2024). Genie: Generative Interactive Environments. ICML.

Chi, C., Feng, S., Du, Y., et al. (2023). Diffusion Policy: Visuomotor Policy Learning via Action Diffusion. Robotics: Science and Systems.

Ho, J., Jain, A., & Abbeel, P. (2020). Denoising Diffusion Probabilistic Models. NeurIPS.

Jang, J., Ye, S., Lin, Z., Xiang, J., Bjorck, J., Fang, Y., Hu, F., Huang, S., et al. (2025). DreamGen: Unlocking Generalization in Robot Learning through Neural Trajectories. arXiv:2505.12705 (NVIDIA GEAR).

Kim, M. J., et al. (2024). OpenVLA: An Open-Source Vision-Language-Action Model.

LeCun, Y. (2022). A Path Towards Autonomous Machine Intelligence. OpenReview.

Lewis, P., Perez, E., Piktus, A., et al. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. NeurIPS.

Liu, B., Zhu, Y., Gao, C., et al. (2023). LIBERO: Benchmarking Knowledge Transfer for Lifelong Robot Learning. NeurIPS Datasets and Benchmarks.

Oquab, M., Darcet, T., Moutakanni, T., et al. (2023). DINOv2: Learning Robust Visual Features without Supervision. TMLR.

Shao, Z., Wang, P., Zhu, Q., et al. (2024). DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. arXiv:2402.03300.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. JMLR 15:1929–1958.

Vaswani, A., Shazeer, N., Parmar, N., et al. (2017). Attention Is All You Need. NeurIPS.

Wu, H., Jing, Y., Cheang, C., et al. (2024). Unleashing Large-Scale Video Generative Pre-training for Visual Robot Manipulation (GR-1). ICLR.

Ye, S., Ge, Y., et al. (2026). DreamZero: World Action Models are Zero-Shot Policies. arXiv:2602.15922.

Zhai, X., Mustafa, B., Kolesnikov, A., & Beyer, L. (2023). Sigmoid Loss for Language Image Pre-Training. ICCV.

Zhu, Y., Wang, J., Mandlekar, A., et al. (2024). Retrieval-Augmented Embodied Agents. CVPR.

Black, K., Brown, N., Driess, D., Esmail, A., Equi, M., et al. (2024). pi0: A Vision-Language-Action Flow Model for General Robot Control. arXiv:2410.24164.

Hafner, D., Pasukonis, J., Ba, J., & Lillicrap, T. (2023). Mastering Diverse Domains through World Models. arXiv:2301.04104.

Hassabis, D., Kumaran, D., Summerfield, C., & Botvinick, M. (2017). Neuroscience-Inspired Artificial Intelligence. Neuron, 95(2), 245-258.

NVIDIA. (2025). Cosmos World Foundation Model Platform for Physical AI. arXiv:2501.03575.

Atance, C. M., & O'Neill, D. K. (2001). Episodic future thinking. Trends in Cognitive Sciences, 5(12), 533-539.

Appendix A. GR-1 Encoder Configuration

We use the public GR-1 checkpoint trained on the Ego4D ABCD split (Wu et al., 2024), together with the MAE ViT-base checkpoint shipped with the GR-1 release. In our pipeline, 128x128 LIBERO RGB frames are resized to 224x224, encoded by the MAE/GR-1 visual stack, and projected to a 384-dimensional observation embedding. GR1Encoder.predict_next_feat calls the frozen GR-1 forward pass to obtain one-step predicted observations; for visualization we additionally keep obs_preds, unpatchify the 14x14 grid of 16x16 RGB patches into a 224x224 image, resize to 128x128, and compare it against the current HDF5 frame. The cached training features use consecutive demonstration transitions and store feat_t, feat_next, action, and transition metadata in per-task .pt files.

Appendix B. Main Experiment: Full Results (GR-1 + RAID, LIBERO-Spatial)

All code, training artifacts, and result files for the main experiment are available at https://github.com/ConstantinVictorBeatErtel/RAID. The final GR-1 visualization and GRPO artifacts were prepared on the codex/visualize-transitions-update branch / PR #1 before merge into main.

B.1 Full Validation MSE Table

The headline results report point estimates from configs/results_libero.json, not three-seed averages. The final sweep evaluates raid_visual and direct_visual across N = 25, 50, 100, and 200 LIBERO-Spatial demonstrations using the same cached GR-1 feature dataset and best-validation-MSE checkpointing.

B.2 Training and Validation Loss Curves

Per-epoch training and validation MSE for each condition and demonstration scale are stored in configs/loss_curves_{N}demos.json. Each file contains train and val loss arrays indexed by epoch for both raid_visual and direct_visual. Training runs for 100 epochs with best-validation-MSE checkpointing; the curves show that raid_visual converges faster and to a lower floor at all scales, with the gap most pronounced at N=25 where the retrieval prior compensates for the paucity of parametric training signal.

B.3 Per-Action-Dimension MSE Breakdown

The 7-DOF action vector comprises (dx, dy, dz, dθ_x, dθ_y, dθ_z, grip). Per-dimension MSE is computed in src/run_all_libero.py alongside the aggregate metric and written to configs/results.json under the key per_dim_mse. RAID's advantage is concentrated in the translational dimensions (dx, dz), where the dreamed next-state feature provides the strongest directional signal. Rotational dimensions (dθ_x, dθ_y, dθ_y) show higher residual error in both heads, consistent with the low rotational diversity of LIBERO-Spatial pick-and-place tasks.

B.4 Attention Gate Analysis

The per-dimension gate g = sigmoid(W concat(f_t, f_{t+1}) + b) controls how much each action dimension is drawn from the direct MLP trunk versus the cross-attended retrieval prior. This gate is implemented in RAIDDecoderVisual as a 7-dimensional output, so different action coordinates can rely on retrieval to different degrees. We did not log mean gate values in the final sweep, so the report treats the gate as an architectural mechanism rather than an analyzed statistic.

B.5 Memory Bank Retrieval Statistics

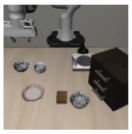
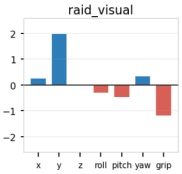
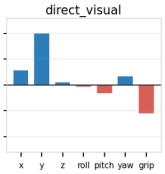
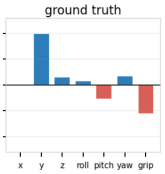
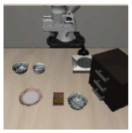
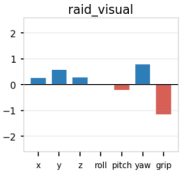
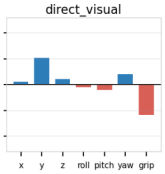
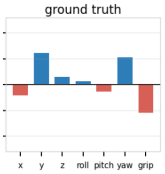
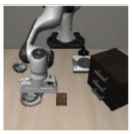
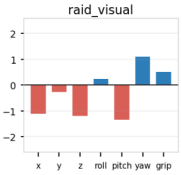
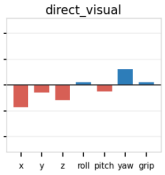
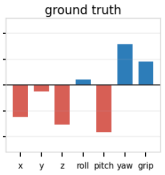
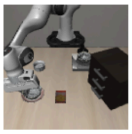
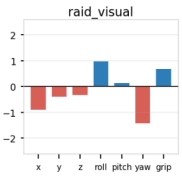
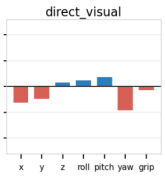
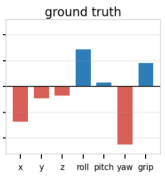
The memory bank implementation holds up to 300,000 train-split transitions. For the final 200-demo LIBERO-Spatial experiment, populate_memory_from_cache builds a 24,733-entry bank from cached GR-1 transitions. Retrieval is exact dense cosine similarity in PyTorch (normalized query-key matrix multiplication followed by torch.topk), not an approximate FAISS index. This should be reproducible from src/memory.py and src/train_libero.py.

B.6 Loss Curves

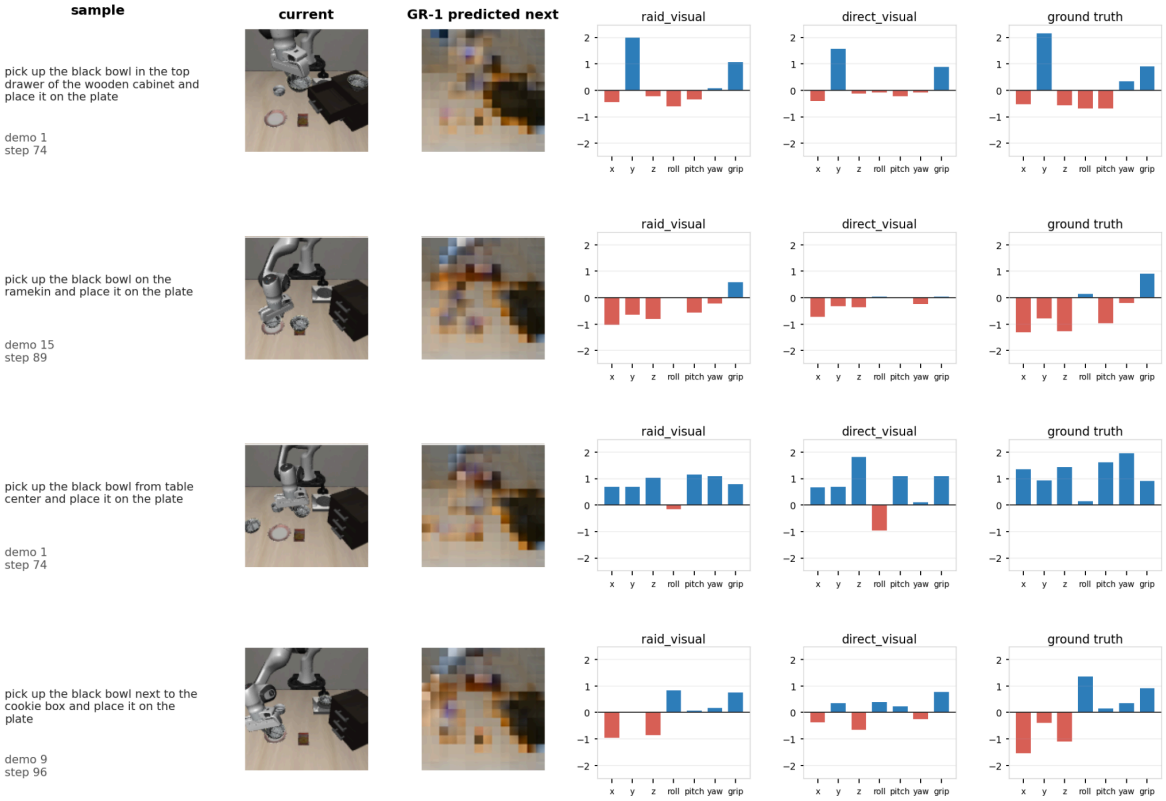
Appendix C. Qualitative Transition Samples

The appendix transition grids are qualitative diagnostics sampled from held-out cached LIBERO-Spatial transitions. Each row displays the current RGB observation, the GR-1-dreamed next observation, and predicted 7-DOF action bar charts for raid_visual, direct_visual, and ground truth. The grids are meant to make model behavior inspectable; they are not additional quantitative evaluation metrics.

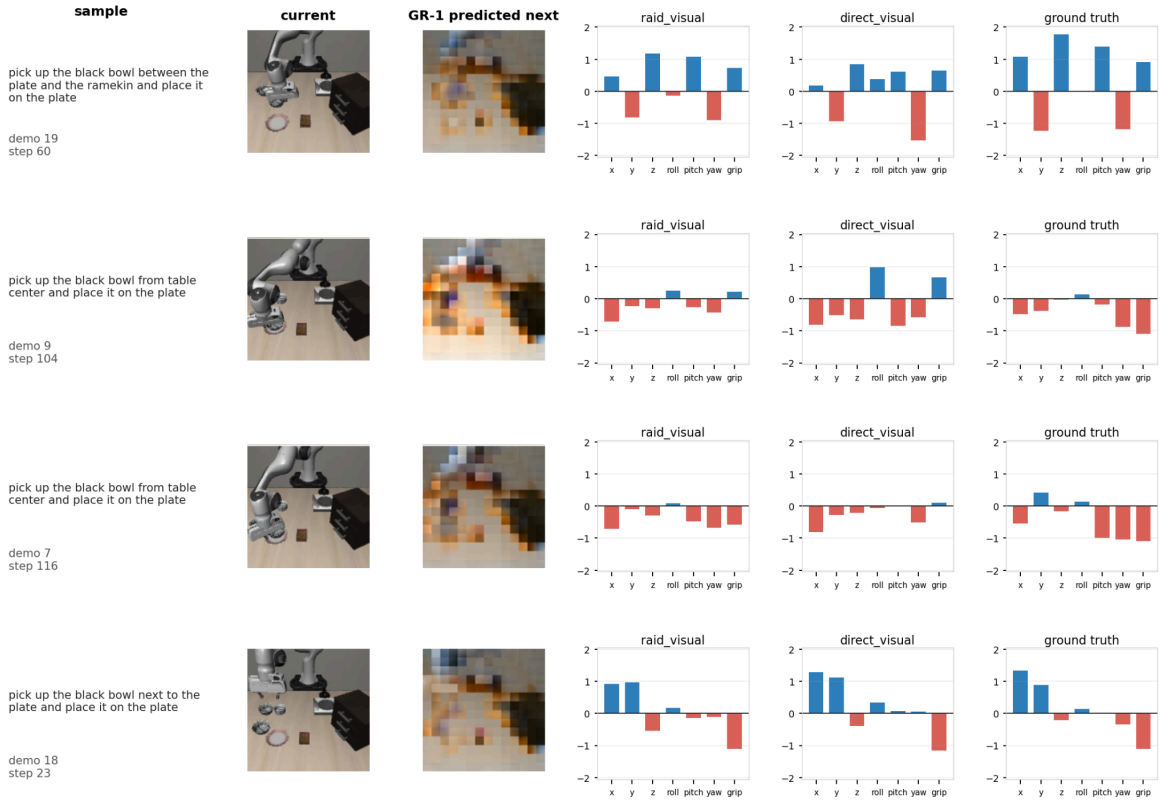
RAID transition samples from cached LIBERO features

sample	current	GR-1 predicted next	raid_visual	direct_visual	ground truth
pick up the black bowl next to the ramekin and place it on the plate demo 4 step 28					
pick up the black bowl next to the ramekin and place it on the plate demo 13 step 1					
pick up the black bowl between the plate and the ramekin and place it on the plate demo 13 step 72					
pick up the black bowl next to the cookie box and place it on the plate demo 10 step 110					

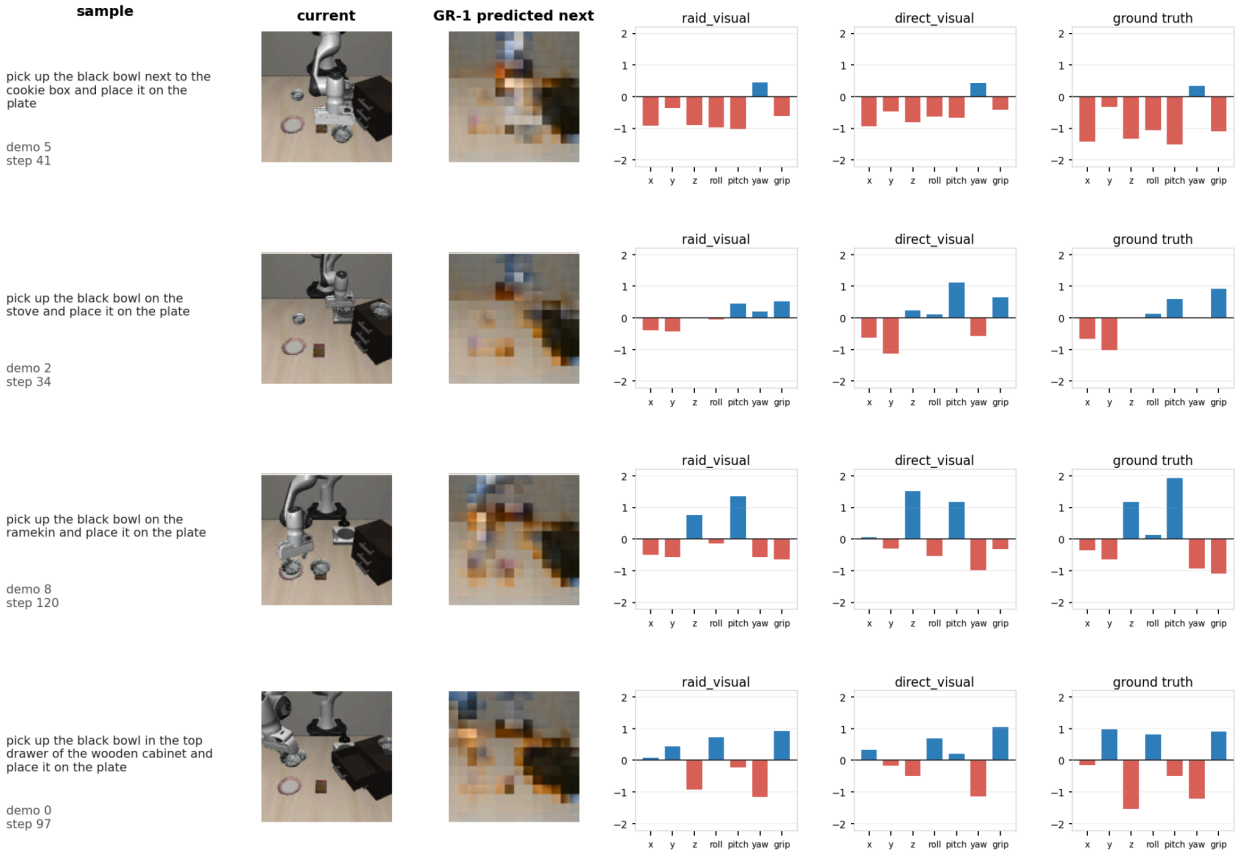
RAID transition samples from cached LIBERO features



RAID transition samples from cached LIBERO features



RAID transition samples from cached LIBERO features



Appendix D. GRPO

D.1 Purpose

The main RAID experiments evaluate inverse dynamics offline through validation MSE on normalized demonstrator actions. This is necessary but incomplete: a low action-prediction error does not guarantee that the policy remains stable when rolled out closed-loop in simulation. We therefore ran a small online fine-tuning probe using Group Relative Policy Optimization (GRPO) to test whether the behavior-cloned GR-1 + RAID policy could be pushed toward actual LIBERO task success.

D.2 Setup

We initialized from the best N = 200 behavior-cloned GR-1 + cross-attention RAID checkpoint. The environment was LIBERO-Spatial task 0: "pick up the black bowl between the plate and the ramekin and place it on the plate." Rollouts used a 150-step horizon, G = 4 rollouts per update, EGL/MuJoCo rendering on a Lambda A10 GPU instance, and a frozen behavior-cloned RAID reference for regularization.

D.3 Rewards

The rollout reward combined the LIBERO environment reward, a sparse success bonus, and dense shaping. The shaping term penalized end-effector distance to the target object and gave a one-time lift bonus when the bowl crossed a height threshold. Rewards were scaled to keep the 150-step horizon comparable to earlier 30-step tests.

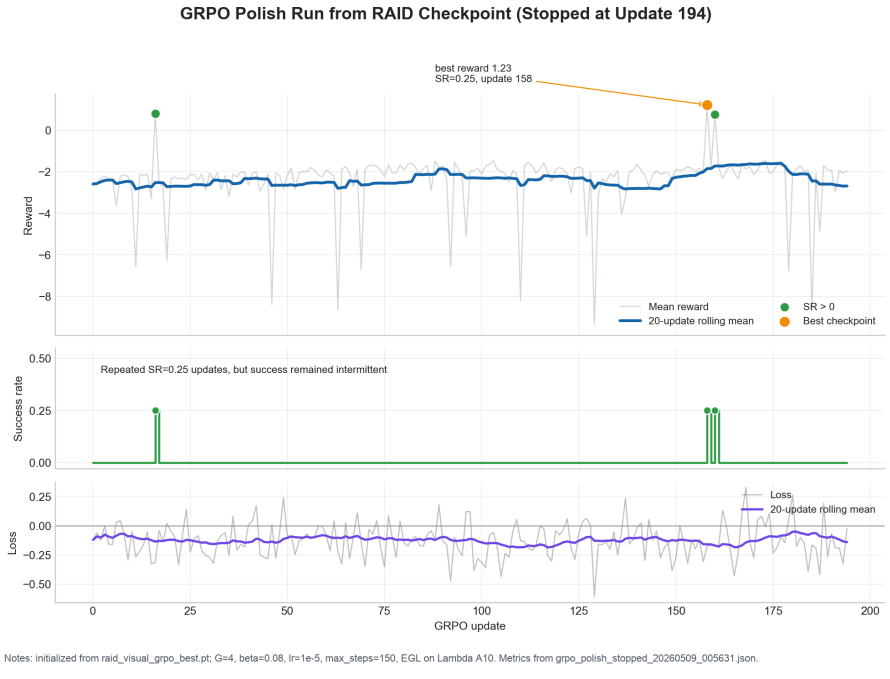
D.4 Implementation Fixes

During debugging, three implementation issues materially affected stability:

- The object-to-end-effector observation key sometimes returned a 6D pose vector rather than a 3D position vector. We fixed reward shaping to compute distance using only the first three coordinates.
- The policy's log_std parameter was unconstrained, causing exploration variance and log probabilities to become unstable when advantages clustered near zero. We added 0.01 * ||log_std||² regularization and clamped log_std to [-3, 0] after each optimizer step.
- The GRPO implementation accumulated gradients over G = 4 rollout losses before stepping the optimizer, but did not divide each loss by G. We corrected this so the effective update scale matched the intended learning rate.

D.5 Results

The GRPO run logged 195 updates and reached best mean reward 1.226 at update 158. It intermittently reached best success rate 0.25 over a group of four rollouts, while the final last-25-update success rate returned to 0. The correct interpretation is partial online improvement from the behavior-cloned RAID initialization, not solved closed-loop manipulation.



D.6 Interpretation

We interpret GRPO as promising but inconclusive. The result shows that RAID is roll-out capable and that online optimization can improve closed-loop shaped reward from the behavior-cloned initialization. The occasional successes are important because they show the policy can reach the sparse terminal condition. However, the lack of stable success suggests that future work should focus on reward design, checkpoint selection, larger rollout groups, and possibly a staged curriculum for grasping before placement.

Appendix E. Pre-Experiment Matrix (DINOv2 / SigLIP / RoboMimic / pooled LIBERO)

These results are kept on the record because they motivated the switch to GR-1, but they are not the headline finding of this report. The full v2 experimental matrix expands to 53 cells across phases A–E and was repeated with three seeds (42, 1337, 2024) for a total of 171 runs.

E.1 Cross-Attention RAID forward pass

```
def forward(self, s_t, s_next, retrieved_actions, kv_key_padding_mask=None):
    x = torch.cat([s_t, s_next], dim=-1)         # (B, 2*obs_dim)
    q = self.q_proj(x).unsqueeze(1)              # (B, 1, d_attn)
    k = self.k_proj(retrieved_actions)           # (B, k, d_attn)
    scores = (q @ k.transpose(-2, -1)) / self._sqrt_d
    if kv_key_padding_mask is not None:
        scores = scores.masked_fill(kv_key_padding_mask.unsqueeze(1), float("-inf"))
    attn_weights = F.softmax(scores, dim=-1)     # (B, 1, k)
    a_prior_attn = (attn_weights @ retrieved_actions).squeeze(1)
    g = torch.sigmoid(self.gate_lin(x))          # (B, action_dim)
    d = self.direct(x)                           # (B, action_dim)
    if self.training:
        ap = self.prior_drop(a_prior_attn) + torch.randn_like(a_prior_attn) *
self._prior_noise_std
    else:
        ap = a_prior_attn
    return g * d + (1.0 - g) * ap, attn_weights
```

E.2 Phase C / Cs — RoboMimic Square with DINOv2

Cross-attention RAID is statistically tied with the direct MLP at native sampling rate (Phase C). With temporal stride 5 (Phase Cs, ≈ 250 ms window at 20 Hz), the visual delta becomes meaningful and cross-attention RAID is the lowest-MSE head at every Cs cell — the qualitative shift that GR-1 features then amplifies in §5.

Phase / Head (Square, DINOv2)	N=10	N=25	N=50	N=200
C / DirectMLP	0.188 ± 0.005	0.100 ± 0.001	0.108 ± 0.002	0.101 ± 0.001

C / RAID (mean-pool)	0.189 ± 0.002	0.101 ± 0.003	0.111 ± 0.004	0.105 ± 0.007
C / RAID cross-attn	0.188 ± 0.006	0.100 ± 0.003	0.108 ± 0.002	0.106 ± 0.001
C / Transformer IDM	0.200 ± 0.001	0.151 ± 0.006	0.118 ± 0.002	0.100 ± 0.001
C / Diffusion-Policy	0.551 ± 0.052	0.123 ± 0.004	0.171 ± 0.008	0.179 ± 0.004
Cs / DirectMLP	—	0.087 ± 0.002	—	0.080 ± 0.000
Cs / RAID (mean-pool)	—	0.088 ± 0.006	—	0.080 ± 0.001
Cs / RAID cross-attn	—	0.083 ± 0.003	—	0.079 ± 0.001
Cs / Transformer IDM	—	0.111 ± 0.003	—	0.079 ± 0.000

Mean ± SD across three seeds.

E.3 Phase E — Pooled LIBERO-Spatial DINOv2 retrieval

Pooling 10 LIBERO-Spatial tasks into one DINOv2 memory bank does not let RAID overtake the direct MLP. This rules out "more memory" as the explanation for the GR-1 win.

Phase E head (LIBERO-Spatial pooled, DINOv2, stride 5)	N=50	N=200
DirectMLP	0.072 ± 0.001	0.079 ± 0.001
RAID (mean-pool)	0.079 ± 0.004	0.081 ± 0.003
RAID cross-attn	0.080 ± 0.003	0.081 ± 0.002

E.4 Phase B — Multi-task RoboMimic low-dim

Head	Mixed full (200 demos)	Mixed subset 25 (200 demos)
direct_mlp	0.127 ± 0.001	0.218 ± 0.002
transformer	0.154 ± 0.004	0.261 ± 0.002
diffusion	1.208 ± 0.116	0.862 ± 0.049

Diffusion-Policy fails outright as a horizon-1 IDM head when the noise floor of the schedule outweighs the regression signal on a small action target.

E.5 Mean-Pool RAID architecture-search log

#	Idea	Val MSE (25 demos)	Decision
0	Baseline (concat decoder)	0.444	—
1	Residual: a_prior + δ	0.622	reverted
2	Detached prior in concat MLP	0.444	superseded

3	Learned per-dim sigmoid gate	0.431	kept
4	Separate transition / prior encoders	0.483	reverted
5	Scalar-scaled prior + transition-only trunk	0.446	reverted
6	+ Prior dropout (p = 0.5)	0.399	kept
7	+ Prior Gaussian noise (σ = 0.1)	0.397	kept (best)
8	2× wider direct trunk	0.410	reverted

Iteration 7 is the accepted mean-pool design; the remaining gap to direct MLP (0.397 vs 0.336) motivated the cross-attention variant.

E.6 Hyperparameters and code

• Optimiser: AdamW, $\beta_1 = 0.9$, $\beta_2 = 0.999$, lr 3×10^{-4}, weight decay 10^{-5}.

• Batch size 256; 50 epochs; best-validation-MSE checkpointing.

• Random seeds 42, 1337, 2024 (3 per cell, mean and SD reported).

• RAID memory: max 50,000 entries, cosine similarity, k = 3.

• Cross-attention head: d_attn = 64, single attention head.

• Code: https://github.com/ConstantinVictorBeatErtel/RAID — branch main (GR-1 + GRPO version); branch jp_branch (the v2 pre-experiment matrix).

Appendix F. Compute

Compute environment. Final remote runs used a Lambda Labs Ubuntu instance with an NVIDIA A10 GPU.

GPU rendering

MuJoCo/LIBERO rollouts used EGL rendering with MUJOCO_GL=egl. We installed libnvidia-gl-580-server and added the ubuntu user to the render and video groups so EGL device access worked without an interactive desktop session.

Software environment

The RAID environment used Python/PyTorch with transformers==4.40.2, flamingo-pytorch, robosuite/LIBERO installed from source at ~/LIBERO, h5py for HDF5 demonstration loading, and matplotlib/Pillow for transition-grid visualizations.

GR-1 compatibility patches

The external GR-1 checkout at ~/GR1 required three compatibility fixes: removing the tokenizer_class keyword from add_code_sample_docstrings, replacing config.n_ctx with config.n_positions for newer transformers, and slicing embed_timestep.weight to the active sequence length because the checkpoint stores 10 timesteps while inference uses seq_len=1.

Data and checkpoints

The LIBERO-Spatial dataset lived at ~/RAID/data/libero_spatial/libero_spatial/libero_spatial with 10 HDF5 task files and 50 demonstrations per task. Cached GR-1 features were stored in ~/RAID/data/libero_spatial/features. GR-1 checkpoints were stored under checkpoints/gr1, and trained behavior-cloning models under models/{raid_visual,direct_visual}_{25,50,100,200}demos_libero_best.pt.

GRPO artifacts

The final bundle raid_grpo_final contains the GRPO scripts, polish-run logs, plots, and recommended checkpoint raid_visual_grpo_polish_best_20260509_005631.pt. The final report discusses only the polished GRPO run: 195 logged updates, best mean reward 1.226 at update 158, and intermittent best success rate 0.25.