DIVER: DECISION-CRITICAL VERIFIER LEARNING FOR VLA TEST-TIME SCALING

Anonymous authors
Paper under double-blind review

ABSTRACT

Scaling robot data and model capacity has improved Vision-Language-Action (VLA) policies, but further progress is constrained by the high cost of robotic data. Verifier-guided test-time scaling offers an efficient alternative by sampling multiple action candidates and selecting the one most likely to lead to task success at inference time. Existing classification-based verifiers learn from trajectory-level outcomes but treat all visited states equally, even though their value for candidate discrimination can vary across a trajectory. At many states, plausible actions are similar and provide limited discrimination signal, while only a sparse set of decision-critical states admits meaningfully different actions that can substantially affect downstream outcomes. To address this, we propose DIVER, which estimates decision criticality from the dispersion of sampled action representations. DIVER then uses this signal to reweight verifier learning toward states where action selection is most consequential, without requiring step-level annotations or additional environment interaction. Across LIBERO, RoboCasa, and real-world experiments on a Franka Research 3 robot, DIVER consistently improves task success through more effective verifier-guided action selection, while adding negligible verifier inference overhead. Videos are available at: Project Page.

1 INTRODUCTION

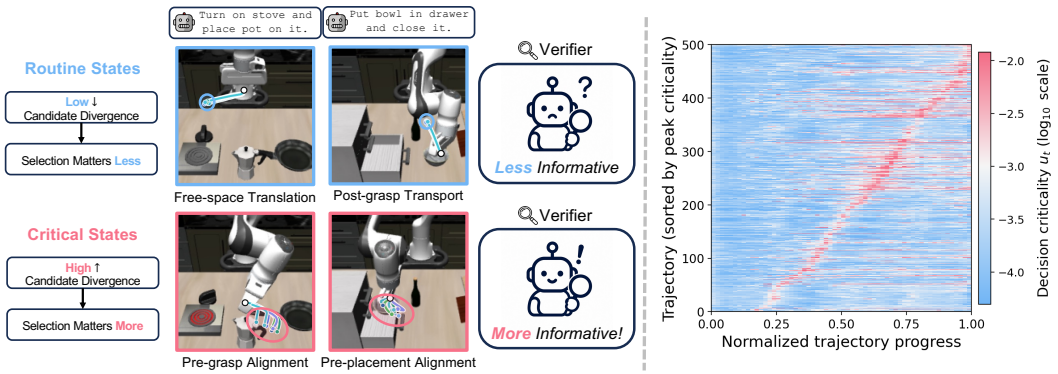
Vision-Language-Action (VLA) models (Kim et al., 2024; Black et al., 2024; Intelligence et al., 2025b;a; Bjorck et al., 2025) learn generalist robot policies from large-scale multimodal demonstrations, but further scaling remains limited by the high cost of robotic data collection (Ma et al., 2026). Test-Time Scaling (TTS)[1] (Snell et al., 2024; Jang et al., 2026; Kwok et al., 2025) offers a complementary route to improving VLA performance without collecting new demonstrations or retraining the base policy. Rather than executing a single sampled action, a pretrained VLA can be viewed as a stochastic generator that proposes multiple candidate action chunks at each decision step. Since this distribution may contain both successful and unsuccessful continuations, a verifier can rank the candidates and select the one most likely to make task progress. This naturally leads to verifier-guided Best-of-N (BoN) sampling, a generator–verifier framework that converts additional inference-time computation into better action selection.

A practical approach to verifier learning is to train lightweight classification-based verifiers (Yang et al., 2025; Tsao et al., 2026; Attarian et al., 2026) from trajectory-level success and failure outcomes, avoiding costly step-level supervision (Zhang et al., 2026) and the inference overhead of external VLM-based judges (Liang et al., 2026; Kwok et al., 2025; 2026a). However, existing approaches propagate each trajectory outcome uniformly across all visited states or state–action pairs, implicitly treating them as equally informative for candidate discrimination. This assumption can be problematic in long horizon robotic execution, where the informativeness of trajectory-level supervision for candidate discrimination can vary substantially across states.

Our key observation is that *not all states are equally consequential for action selection*. As illustrated in Figure 1(a), at states such as free-space motion or post-grasp transport, sampled action candidates remain highly similar. We refer to these as **routine states**, where candidate selection is relatively inconsequential and provides limited signal for learning to discriminate among actions. In contrast, at

[1]We use test-time scaling to denote parallel scaling via repeated sampling and ranking multiple action candidates, rather than sequential scaling via extended rollouts or iterative refinement (Madaan et al., 2023).

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.



(a) Routine vs. Decision-Critical States (b) Sparsity of Decision-Critical States

Figure 1: **Not all states are equally informative for verifier learning.** (a) At routine states, sampled action candidates remain similar, so distinguishing among them is less important for downstream behavior. At decision-critical states, candidates diverge more strongly, making accurate selection more consequential for task outcomes. (b) Across 500 LIBERO-Long trajectories from a frozen π_0 policy, states with high decision criticality, measured by u_t (Equation (8)), are sparse and temporally localized.

decision-critical states, such as pre-grasp or pre-placement alignment, plausible candidates diverge more strongly, and small differences in action choice can substantially affect downstream outcomes. Figure 1(b) further shows that such high-criticality states are sparse and temporally localized rather than uniformly distributed throughout a trajectory. This pattern echoes findings in language generation, where only a sparse subset of high-entropy minority tokens can lead to divergent continuations and steer generation toward distinct downstream trajectories (Wang et al., 2025). Together, these observations suggest that uniformly weighting all visited states can dilute the supervision signal from the relatively few states where accurate candidate discrimination matters most.

In this paper, we introduce DIVER, a **D**ecision-cr**i**ticality-weighted **VeR**ifier for VLA test-time scaling that focuses learning on states where action selection matters most. Our key idea is to estimate decision criticality from disagreement among plausible actions under the frozen VLA policy. At each visited state, we sample multiple action candidates and measure the dispersion of their action-expert representations. Low dispersion characterizes routine states where the policy proposes similar behaviors, whereas high dispersion identifies states where plausible actions diverge and candidate selection can substantially affect downstream outcomes. This provides a practical proxy for decision criticality without requiring step-level supervision or additional environment interaction. DIVER uses this signal to reweight the verifier objective, concentrating trajectory-level supervision on states where accurate candidate discrimination is most consequential.

We evaluate DIVER on two simulated manipulation benchmarks and a real-world robotic platform using π_0 and $\pi_{0.5}$. Across policies and environments, DIVER improves task success by up to **+13.6%** over single sample inference and **+3.4%** over the strongest baseline. Notably, DIVER also surpasses the VLM-based verifier by **+4.6%** success rate while reducing verifier inference latency by over **700**×. These results demonstrate the effectiveness of focusing verifier learning on the sparse states where candidate differences matter most.

Our key contributions are summarized as follows:

- We identify a limitation of uniform verifier training, showing that routine states dominate trajectories while providing limited signal for candidate discrimination, whereas sparse decision-critical states are more consequential for action selection.
- We propose DIVER, a novel decision-criticality-weighted verifier that uses action-expert representation variance as a practical proxy for decision criticality and places greater emphasis on states with stronger candidate disagreement.
- We validate DIVER across simulated and real-world robotic manipulation tasks, demonstrating consistent improvements in task success through more effective verifier-guided Best-of-N action selection, with negligible verifier overhead at inference.

2 Related Works

Vision-Language-Action (VLA) models learn generalist robot policies by fine-tuning pretrained vision-language backbones on large-scale robot demonstration data (Ma et al., 2026; Kim et al., 2024; Bjorck et al., 2025; Black et al., 2024; Barreiros et al., 2025). Recent approaches further improve these policies along several complementary directions, including geometry-aware representations that enhance 3D spatial understanding (Qu et al., 2025; Li et al., 2026), memory mechanisms for long-horizon tasks (Shi et al., 2026; Torne et al., 2026; Sridhar et al., 2026), and world models that capture environment dynamics (Ye et al., 2026; Cen et al., 2025; Gao et al., 2026). While effective, these directions strengthen the policy itself and thus require additional data collection, architectural modification, or retraining. In contrast, we treat the pretrained VLA as a frozen stochastic action generator and improve its execution purely at inference time, making our approach complementary to these advances.

Test-time Scaling in LLMs improves performance by allocating additional computation at inference time rather than through further training, e.g., via repeated sampling with majority voting (Wang et al., 2023; Brown et al., 2024; Kang et al., 2025) or verifier-guided Best-of-N selection with outcome- and process-level reward models (Zhang et al., 2025; Snell et al., 2024). Recent studies further suggest that not all generation steps are equally consequential: a small subset of high-entropy *forking tokens* can act as branching points that steer reasoning toward different continuations, while many low-entropy tokens largely follow an already determined trajectory (Bigelow et al., 2025; Wang et al., 2025; Al-Khalili et al., 2026). Focusing supervision on these forking tokens can therefore yield disproportionate gains, suggesting that the value of supervision may vary across a trajectory. We study an analogous non-uniformity in robotic control through *decision-critical states*, where candidate action selection can have substantially different downstream consequences. Accordingly, we focus verifier learning on these sparse states rather than treating all states uniformly.

Test-time Scaling in VLAs. Motivated by advances in LLM test-time scaling, recent works have adopted generate-then-verify paradigms for robotic control. For example, (1) VLM-based verifiers score candidate actions with large vision-language models (Kwok et al., 2025; 2026b; Dai et al., 2025; Liang et al., 2026; Duan et al., 2025), but their inference cost and deployment overhead hinder real-time control; (2) Verifier-free methods instead rely on the policy's internal signals, such as distributional confidence or candidate density (Jang et al., 2026; Rosasco et al., 2025; Choi et al., 2026), yet these signals are not grounded in task outcomes; (3) Classification-based verifiers (Yang et al., 2025; Attarian et al., 2026; Tsao et al., 2026) offer a lightweight alternative by learning from trajectory-level outcomes, but weight all visited states uniformly. Our approach instead accounts for the non-uniform importance of states by estimating decision criticality from action-expert representation variance and placing greater training emphasis on states where candidate actions diverge and selection is most consequential.

3 Problem Setup

Verifier-guided Best-of-N action selection. We consider a generator–verifier framework for improving a generalist VLA policy through test-time compute. We model robotic execution over decision steps $t \in [T]$, with policy-conditioning state space $\mathcal{S}$ and action space $\mathcal{A}$, where T is the trajectory horizon. The state $s_t \in \mathcal{S}$ includes the current visual observation, language instruction, and any additional context used by the policy. Conditioned on s_t, the pretrained VLA policy $\pi_{\text{pre}}(\cdot \mid s_t)$ generates an action chunk $\mathbf{a}_t = (a_{t,1}, \ldots, a_{t,H}) \in \mathcal{A}^H$, where H is the action-chunk horizon. Rather than executing a single prediction, we treat the policy as a stochastic generator and sample N candidate action chunks,

$$\mathbf{a}_t^{(i)} \sim \pi_{\text{pre}}(\cdot \mid s_t), \qquad i \in [N]. \tag{1}$$

A learned verifier $\widehat{f} : \mathcal{S} \times \mathcal{A}^H \to \mathbb{R}$ assigns a scalar score to each candidate, and the robot executes

$$\mathbf{a}_t^\star = \arg\max_{i \in [N]} \widehat{f}(s_t, \mathbf{a}_t^{(i)}). \tag{2}$$

Thus, the VLA policy serves as a stochastic proposer of plausible robot behaviors, while the verifier provides an inference-time selection rule for choosing the candidate most likely to satisfy the instruction.

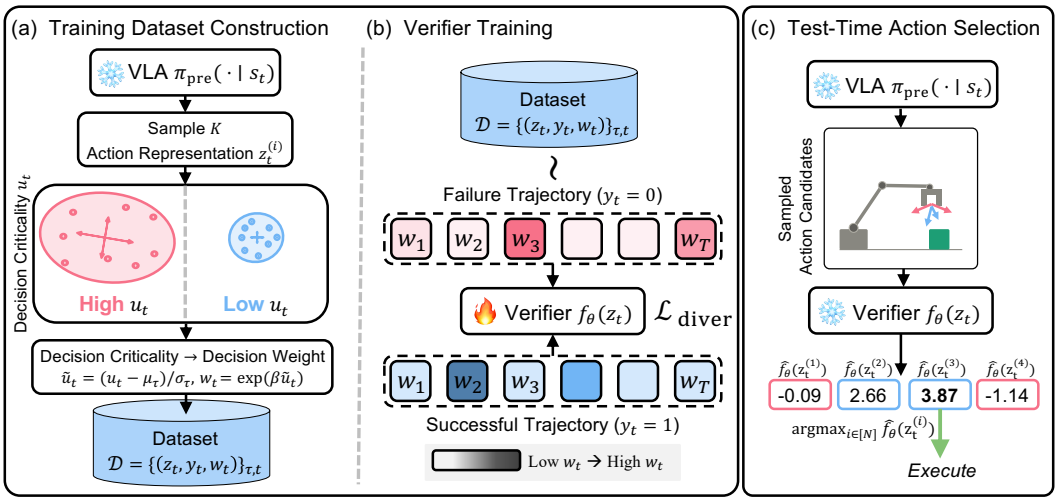


Figure 2: **Overview of DIVER.** (a) At each visited state, we sample K action candidates from the frozen VLA policy and estimate decision criticality u_t from the dispersion of their action-expert representations. The scores are standardized within each trajectory and converted into decision weights w_t, which are paired with the executed action representation z_t and trajectory outcome y_t to construct the training dataset $\mathcal{D}$. (b) The verifier is trained with decision-weighted binary cross-entropy, assigning greater importance to states with larger w_t. (c) At inference, the verifier scores N sampled action candidates at each decision step and executes the highest-scoring action chunk.

4 METHOD

Overview. DIVER learns a lightweight verifier from successful and failed trajectories while emphasizing states where candidate action selection is most consequential. We estimate decision criticality from the dispersion of action-expert representations sampled from the frozen VLA policy and use this signal to weight each state's contribution to verifier learning. The overall framework is illustrated in Figure 2.

4.1 SUCCESS-VISITATION VERIFIER

We formulate the verifier as a lightweight discriminator using trajectory-level success and failure outcomes. Each trajectory provides a sequence of state–action pairs together with a terminal outcome label. Let $\rho^+(s, \mathbf{a})$ and $\rho^-(s, \mathbf{a})$ denote the state–action visitation densities induced by successful and failed trajectories, respectively. The discriminator $f_\theta(s, \mathbf{a}) \in (0, 1)$ distinguishes samples from these two visitation distributions and is optimized using binary cross-entropy:

$$\mathcal{L}_{\text{BCE}}(\theta) = -\mathbb{E}_{(s,\mathbf{a})\sim\rho^+}\left[\log f_\theta(s, \mathbf{a})\right] - \mathbb{E}_{(s,\mathbf{a})\sim\rho^-}\left[\log(1 - f_\theta(s, \mathbf{a}))\right]. \tag{3}$$

Under balanced sampling between successful and failed trajectories, the population-optimal discriminator is

$$f^\star(s, \mathbf{a}) = \frac{\rho^+(s, \mathbf{a})}{\rho^+(s, \mathbf{a}) + \rho^-(s, \mathbf{a})}. \tag{4}$$

Its logit therefore satisfies

$$\log \frac{f^\star(s, \mathbf{a})}{1 - f^\star(s, \mathbf{a})} = \log \frac{\rho^+(s, \mathbf{a})}{\rho^-(s, \mathbf{a})}. \tag{5}$$

Thus, at the population optimum, the discriminator logit recovers the log ratio between successful and failed state–action visitation densities. We define $R(s, \mathbf{a}) = \frac{\rho^+(s,\mathbf{a})}{\rho^-(s,\mathbf{a})}$ as the success-visitation ratio. A larger $R(s, \mathbf{a})$ indicates that a state–action pair is more characteristic of successful than failed trajectories, providing a natural signal for candidate ranking. Accordingly, the discriminator

logit serves as the verifier score:

$$\hat{f}_\theta(s, \mathbf{a}) := \log \frac{f_\theta(s, \mathbf{a})}{1 - f_\theta(s, \mathbf{a})}. \tag{6}$$

In practice, rather than operating directly on raw state–action pairs, we represent each candidate using an internal hidden representation $z_t \in \mathbb{R}^d$ from the VLA policy's action expert (Gu et al., 2025; Yang et al., 2025). We parameterize the discriminator as $f_\theta(z_t)$ and use its logit as the verifier score. In this representation space, the same formulation estimates the log density ratio between representations induced by successful and failed trajectories.

Do all states matter equally? Although this formulation provides a lightweight outcome-grounded verifier, it treats all visited states equally during training. However, the usefulness of each state for candidate discrimination can vary along a trajectory. At **routine states**, successful and failed trajectories often visit similar states and execute similar actions, providing limited discrimination signal. In contrast, at **decision-critical states**, plausible actions can diverge and lead to substantially different downstream outcomes. Because routine states dominate trajectories (Figure 1(b)), uniform weighting can dilute the signal from the sparse states where accurate candidate selection matters most. This motivates estimating decision criticality and placing greater emphasis on these consequential states during verifier learning.

4.2 Decision Criticality from Action-Representation Variance

We seek a practical measure of decision criticality that captures how strongly the policy's plausible actions diverge at a given state, without requiring step-level supervision or additional environment interaction. As illustrated by the examples in Figure 3, candidate action representations can remain tightly concentrated at routine states while becoming more dispersed at decision-critical states. We therefore use candidate disagreement in the frozen VLA policy's internal action-representation space as a proxy for decision criticality. For each state s_t, we sample K candidate action chunks from the frozen policy,

$$\tilde{\mathbf{a}}_t^{(k)} \sim \pi_{\text{pre}}(\cdot \mid s_t), \qquad k \in [K], \tag{7}$$

and extract the corresponding internal hidden representation from the VLA action expert[2] $z_t^{(k)} \in \mathbb{R}^d$, where $z_{t,j}^{(k)}$ denotes its j-th coordinate. We quantify candidate disagreement by the mean per-coordinate variance,

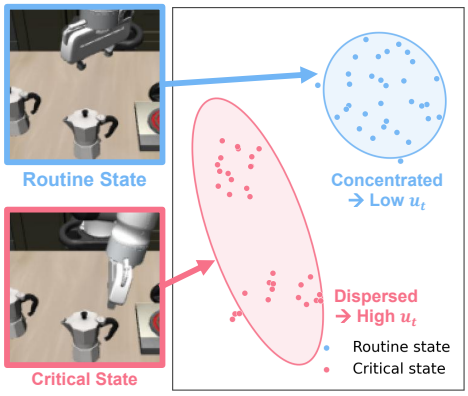


Figure 3: **Sampled action representations.** Candidate representations are concentrated at a routine state (e.g., free-space motion) but more dispersed at a decision-critical state (e.g., before grasping).

$$u_t = \frac{1}{d} \sum_{j=1}^{d} \mathrm{Var}_{k \in [K]} \left[z_{t,j}^{(k)} \right] = \frac{1}{d} \mathrm{Tr} \left(\mathrm{Cov}_{k \in [K]} \left[z_t^{(k)} \right] \right). \tag{8}$$

We use u_t as the decision-criticality score. Lower values of u_t indicate more concentrated candidate actions, characterizing routine states where there is limited opportunity for meaningful candidate discrimination. Conversely, higher values reflect stronger disagreement among plausible actions, serving as a proxy for decision-critical states where accurate candidate selection may be more consequential. We further verify in Appendix C.1 that u_t correlates with rollout-based oracle criticality.

4.3 DiVeR: Decision-Weighted Verifier Learning

Decision-weighted training dataset. Let $\mathcal{T}$ denote the set of offline trajectories collected by the frozen VLA policy. Each trajectory $\tau \in \mathcal{T}$ consists of a sequence of visited states and executed actions with a terminal success or failure label. For each timestep t in trajectory τ, we store the

[2]Details of the representation extraction are provided in Appendix B.

action-expert representation z_t of the executed action chunk and compute its decision-criticality score u_t from K sampled candidates according to Equation (8). To obtain a relative measure of criticality within each trajectory, we standardize the scores and convert them into positive decision weights using exponentiated weighting (Peng et al., 2019):

$$\tilde{u}_t = \frac{u_t - \mu_\tau}{\sigma_\tau}, \quad w_t = \exp(\beta \tilde{u}_t), \quad \beta \geq 0, \tag{9}$$

where μ_τ and σ_τ denote the mean and standard deviation of the criticality scores in trajectory τ, and β is the weight temperature. This transformation assigns larger weights to states with above-average candidate disagreement while downweighting more routine states. The resulting verifier-training dataset is

$$\mathcal{D} = \{(z_t, y_t, w_t)\}_{\tau \in \mathcal{T},\, t \in [T_\tau]}, \tag{10}$$

where T_τ denotes the length of trajectory τ, and $y_t \in \{0, 1\}$ denotes the trajectory-level outcome propagated to timestep t, with $y_t = 1$ for success and $y_t = 0$ for failure.

Decision-weighted verifier objective. We train the verifier using weighted binary cross-entropy:

$$\mathcal{L}_{\text{diver}}(\theta) = -\mathbb{E}_{(z_t, y_t, w_t) \sim \mathcal{D}} \left[w_t \left(y_t \log f_\theta(z_t) + (1 - y_t) \log \left(1 - f_\theta(z_t)\right)\right)\right]. \tag{11}$$

The decision weight w_t is computed solely from candidate disagreement under the frozen policy and does not directly use the trajectory outcome y_t. The weighting changes the relative contribution of training samples, emphasizing states with greater candidate disagreement where candidate discrimination is potentially more useful. The overall algorithm is summarized in Algorithm 1.

5 Experiments

5.1 Experimental Setup

We evaluate our method in two simulated environments and on a real-world robotic platform. Across all environments, we adopt a *multi-task setting*, training and evaluating *a single verifier shared across all tasks* within each benchmark. Results are averaged over three random seeds. Implementation details are provided in Appendix B.

Simulation environments. We evaluate on **LIBERO-Long** (Liu et al., 2023), which comprises 10 long-horizon manipulation tasks with 50 episodes per task, and **RoboCasa** (Nasiriany et al., 2026), which contains 18 atomic manipulation tasks with 50 episodes per task. For both environments, we use 30 episodes per task for training and the remaining 20 episodes for evaluation.

Real-world environment. We conduct real-world experiments on a **Franka Research 3** across four manipulation tasks. We fine-tune $\pi_{0.5}$ with LoRA (Hu et al., 2022) using 40 human teleoperation demonstrations per task and evaluate on 24 episodes per task. The tasks are: (1) placing a green block in a bowl, (2) placing a cup on a plate, (3) stacking the red block on the blue block, and (4) closing a drawer.

Models. We use the representative flow-based VLA models π_0 (Black et al., 2024) and $\pi_{0.5}$ (Intelligence et al., 2025b) as our primary evaluation policies. Note that DiVeR only requires the policy to generate multiple action candidates and expose an internal action representation, making it applicable beyond flow-matching architectures. We additionally evaluate the autoregressive OpenVLA (Kim et al., 2024) in Appendix C.2. For the flow-based models, we generate candidates by independently sampling the initial Gaussian noise while keeping the policy conditioning fixed, and reuse VLM computation through KV caching. We report action-sampling latency in Appendix C.3.

Baselines. We compare our method with two verifier-free selection methods and two learned verifiers under the same Best-of-N setting, using identical candidate sets sampled from the same frozen policy. **KDPE** (Rosasco et al., 2025) selects the candidate with the highest kernel-density estimate, while **MG-Select** (Jang et al., 2026) ranks candidates using condition-masking confidence. **TACO** (Yang et al., 2025) learns a pseudo-count-based score that favors in-distribution state–action pairs, whereas **SVM** (Tsao et al., 2026) discriminates successful from failed visitation using trajectory-level outcomes. Following the original implementation, we concatenate a CNN-based representation of the current image observation with the corresponding action chunk and train an MLP discriminator on the resulting feature. Mathematical details of the baselines are provided in Appendix E.

Table 1: Average success rates (%) on LIBERO-Long with varying numbers of action candidates N sampled at test time. **Bold** and underlined values denote the best and second-best results, respectively.

Method	Policy: π_0				Policy: $\pi_{0.5}$			
	$N=4$	$N=8$	$N=16$	$N=32$	$N=4$	$N=8$	$N=16$	$N=32$
Baseline	83.5				93.5			
KDPE (Rosasco et al., 2025)	83.5	84.5	82.8	82.3	94.0	92.7	94.5	95.7
MG-Select (Jang et al., 2026)	82.0	80.8	83.3	86.3	94.2	95.0	96.5	96.8
TACO (Yang et al., 2025)	**84.2**	83.0	83.5	85.0	94.5	94.8	95.5	95.7
SVM (Tsao et al., 2026)	82.8	85.2	85.5	86.5	94.0	95.5	94.0	97.0
Ours	$84.0_{\pm 0.5}$	$\mathbf{86.3}_{\pm 1.3}$	$\mathbf{87.5}_{\pm 1.5}$	$\mathbf{89.3}_{\pm 0.3}$	$\mathbf{94.8}_{\pm 0.3}$	$\mathbf{96.0}_{\pm 0.5}$	$\mathbf{97.0}_{\pm 0.5}$	$\mathbf{97.5}_{\pm 0.5}$

Table 2: Category-wise average success rates (%) on RoboCasa atomic tasks. **Bold** and underlined values denote the best and second-best results, respectively.

Method	Policy: π_0					Policy: $\pi_{0.5}$				
	Articulated	Control	Object-Centric	Navigation	Average	Articulated	Control	Object-Centric	Navigation	Average
Baseline	37.2	16.2	42.5	0.0	32.2	50.0	18.8	58.3	0.0	43.1
KDPE	32.8	26.2	**49.2**	0.0	35.0	47.8	20.0	57.5	0.0	43.3
MG-Select	29.3	22.5	36.7	0.0	28.6	53.6	33.8	**66.7**	**15.0**	51.3
TACO	37.1	28.8	35.8	0.0	32.7	52.1	26.2	53.3	0.0	43.9
SVM	37.2	26.2	40.8	0.0	33.9	55.0	20.0	60.8	**15.0**	46.9
Ours	$\mathbf{38.6}_{\pm 1.2}$	$\mathbf{33.8}_{\pm 1.0}$	$46.7_{\pm 1.6}$	$0.0_{\pm 0.0}$	$\mathbf{38.1}_{\pm 1.2}$	$\mathbf{59.3}_{\pm 0.9}$	$\mathbf{41.2}_{\pm 1.7}$	$65.0_{\pm 1.0}$	$\mathbf{15.0}_{\pm 0.0}$	$\mathbf{54.7}_{\pm 1.0}$

5.2 Main Experiments

Simulation environment. In Table 1, we compare our method with verifier-free and verifier-based baselines on LIBERO-Long while varying the number of action candidates sampled at test time. Our method achieves robust performance across both models and candidate budgets. For π_0 with $N = 32$, our method improves the success rate by +5.8% over the $N = 1$ baseline and by +2.8% over the best-performing baseline. In Table 2, we report category-wise performance on RoboCasa atomic tasks with $N = 16$. Our method improves performance across diverse task categories, rather than being biased toward a particular type of task. On average, it improves success rate by **+5.9%** for π_0 and **+11.6%** for $\pi_{0.5}$ over the corresponding $N = 1$ baselines. These results demonstrate the effectiveness of our method across different policies, candidate budgets, and task settings.

Real-robot platform. As shown in Table 3, we evaluate on a Franka Research 3 robot with $N = 16$ and compare DiVeR against the $N = 1$ $\pi_{0.5}$ policy (Base) and a uniformly trained verifier (Uniform). Across all four tasks, DiVeR consistently improves performance, achieving an average success rate of $\mathbf{71.9}\%$, compared with 65.6% for Uniform and 58.3% for Base. This corresponds to gains of $+\mathbf{6.3}$ and $+\mathbf{13.6}$ percentage points, respectively, demonstrating that decision-weighted verifier learning transfers effectively to real-world robotic manipulation.

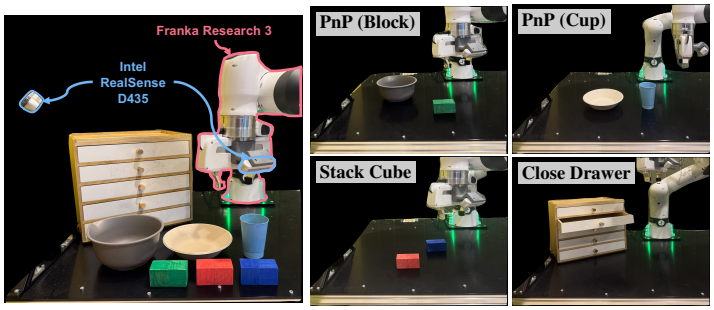
5.3 Quantitative Analysis

In this section, we provide quantitative analyses of DiVeR. Unless otherwise specified, all analyses are conducted on RoboCasa with $N = 16$ and report success rates (%). Additional analyses and ablations are provided in Appendix C.

Component analysis. We ablate the main components of our method by comparing: (i) the base policy with a single sample ($N = 1$), denoted *Baseline*; (ii) random selection among N sampled candidates (*Random selection*); (iii) our verifier trained on action-expert representations with uniform weighting (*Uniform training*); and (iv) the full decision-criticality-weighted verifier (*Ours*).

As shown in Table 4, *random selection* provides only marginal gains over the $N = 1$ baseline, indicating that additional samples are useful only when paired with an effective selection rule. *Training the verifier uniformly* on action-expert representations improves performance by $+2.7$ and $+7.5$ percentage points over the baseline for π_0 and $\pi_{0.5}$, respectively. This variant can be viewed as SVM with its image and raw action input replaced by action-expert representations; compared with the original SVM in Table 2, it further improves performance by $+1.0$ and $+3.7$ points, highlighting the

Table 3: Real-robot evaluation with $\pi_{0.5}$. **Left**: robot setup. **Center**: the four evaluation tasks. **Right**: task success rates (%) over 24 evaluation episodes per task.



Task	Base	Uniform	Ours
PnP (block)	62.5	66.7	**75.0**
PnP (cup)	58.3	66.7	**70.8**
Stack block	45.8	58.3	**66.7**
Close drawer	66.7	70.8	**75.0**
Average	58.3	65.6	**71.9**

Table 4: Component analysis.

Method	π_0	$\pi_{0.5}$
Baseline ($N = 1$)	32.2	43.1
Random selection	33.0	44.1
Uniform training	34.9	50.6
Ours	38.1	54.7

Table 5: Verifier input ablation.

Input feature	π_0	$\pi_{0.5}$
Image & Raw	32.4	50.2
Image & Repr.	37.4	54.6
Representation	38.1	54.7

Table 6: Comparison with a VLM-based verifier.

Method	SR (%) ↑	Latency (s) ↓
RoboMeter	50.1	0.743
Ours	54.7	0.001

benefit of the representation. Finally, decision-criticality weighting (*Ours*) yields an additional $+3.2$ and $+4.1$ points over uniform training, achieving the best overall performance. These results show that both the action-expert representation and decision-criticality weighting contribute to effective candidate selection.

Input feature ablation. In Table 5, we compare verifier inputs based on raw actions, image features combined with action-expert representations, and action-expert representations alone. Using action-expert representations consistently outperforms raw-action inputs, while additionally incorporating image features provides no further benefit. This suggests that the action-expert representation already provides a sufficiently informative, policy-conditioned feature space for candidate ranking. We therefore use action-expert representations alone throughout our method, which also keeps the verifier lightweight and avoids additional visual feature processing.

DiVeR is more effective and efficient than a VLM-based verifier. Using $\pi_{0.5}$, we compare DiVeR with RoboMeter (Liang et al., 2026), a VLM-based reward model for predicting task progress and trajectory preferences, with candidate visual trajectories obtained via simulator branching. As shown in Table 6, DiVeR improves the success rate by $+\mathbf{4.6}$ percentage points while reducing verification latency from 0.743 s to 0.001 s, making the verifier over $\mathbf{700}\times$ faster. These results demonstrate that our lightweight action-representation verifier achieves more effective candidate selection with substantially lower inference overhead.

Can DiVeR generalize to unseen tasks?

We evaluate zero-shot task generalization on RoboCasa with $\pi_{0.5}$ by grouping the manipulation tasks into three functional categories: articulated tasks (e.g., opening or closing), control tasks (e.g., turning a lever), and object-centric tasks (e.g., pick-and-place). We exclude navigation tasks due to their qualitatively different behaviors. For each source category, we train a verifier on its training tasks and evaluate it on held-out tasks from all three categories. In Figure 5, each entry reports the success rate (%) for a training–evaluation category pair; diagonal entries correspond to evaluation within the training category, while off-diagonal entries measure cross-category transfer. As shown, DiVeR transfers effectively across task categories, indicating that the learned verifier generalizes beyond the tasks used for training.

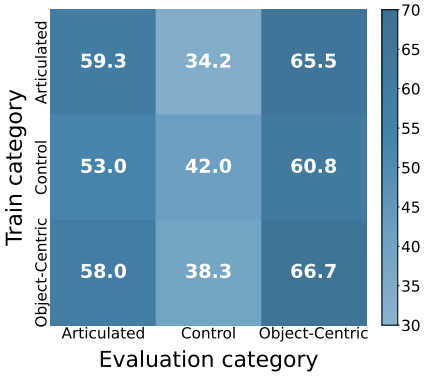
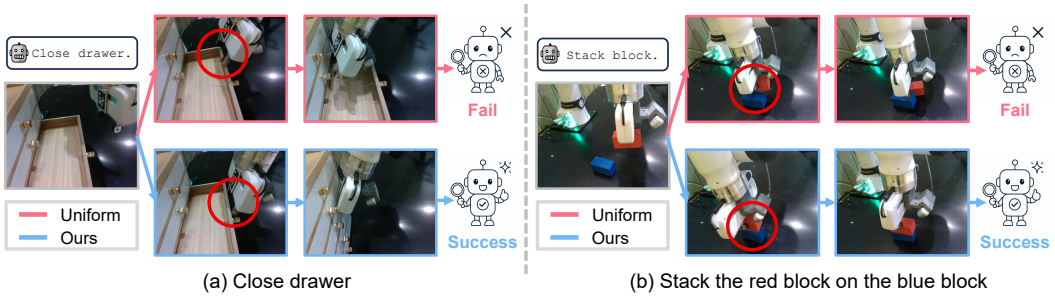


Figure 5: **Cross-task generalization.** Success rates (%) when the verifier is trained on one task category and evaluated on another.



(a) Close drawer (b) Stack the red block on the blue block

Figure 4: **Qualitative comparison of verifier-guided action selection.** Starting from the same decision-critical state, we compare trajectories produced by a uniformly trained verifier and DIVER (Ours). In both (a) drawer closing and (b) block stacking, the two verifiers select different action candidates from the same state, leading to different downstream outcomes. Uniform verifier training selects actions that result in task failure, whereas DIVER selects more appropriate candidates and successfully completes the task. Red circles highlight the resulting behaviors after these action selections.

Verifier architecture. In Figure 6, we compare MLP, LSTM, and single-layer transformer (Vaswani et al., 2017) verifiers under a matched parameter budget.

All architectures use the same action-expert features, training data, and optimization protocol, with approximately 66K trainable parameters each. The MLP performs best across both VLA policies despite its simpler architecture. This suggests that the frozen action-expert representation provides sufficient information for candidate discrimination without requiring additional recurrent or attention-based modeling. We therefore use the MLP as our default verifier due to its strong performance and low inference cost.

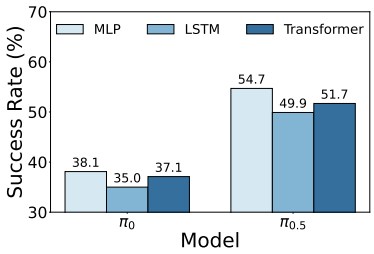


Figure 6: **Verifier architecture.**

5.4 Qualitative Analysis

Action selection at decision-critical states. We further compare the behaviors induced by a uniformly trained verifier and DIVER from the same decision-critical states on a real-world Franka Research 3 robot. As shown in Figure 4, the two verifiers select different action candidates from the same state, leading to different downstream outcomes. (a) For drawer closing, the uniform verifier positions the gripper slightly above the drawer front, causing it to move through free space without making effective contact and ultimately fail. In contrast, DIVER selects an action that accurately positions the gripper against the front of the drawer, successfully completing the closing motion. (b) Similarly, in the block-stacking task, the uniform verifier places the red block slightly beside the blue block rather than on top of it, causing the red block to fall next to the blue block and resulting in task failure. However, DIVER positions the grasped red block directly above the blue block and successfully completes the task. These examples qualitatively illustrate how emphasizing decision-critical states during verifier training can improve candidate selection when action choice is consequential for task completion.

6 Conclusion

We introduced DIVER, a decision-criticality-weighted verifier for VLA test-time scaling. DIVER estimates decision criticality from the dispersion of sampled action representations and places greater training emphasis on sparse states where candidate selection is most consequential. Across LIBERO-Long, RoboCasa, and real-world manipulation tasks, DIVER consistently improves task success over verifier-free and learned baselines through more effective verifier-guided Best-of-N action selection, while adding negligible verifier inference overhead. These results suggest that effective VLA test-time scaling benefits not only from generating more action candidates, but also from focusing verifier learning on the states where their differences matter most.

ADDITIONAL STATEMENTS

AI use statement. In this work, we used generative AI tools to assist with running experiments, including troubleshooting experimental code and workflows. We have not used generative AI tools for other tasks requiring disclosure, and the remaining required disclosure tasks are not applicable to this work. Additionally, we used generative AI tools to edit the manuscript for grammar and readability. We have reviewed and verified all AI-assisted work. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

Ethics statement. This work studies verifier-guided action selection for VLA-based robotic manipulation in simulation and controlled real-world environments. The experiments do not involve human subjects or personally identifiable information. Although our method can improve action selection, it does not provide formal safety guarantees and should be carefully evaluated before use in safety-critical applications.

Reproducibility statement. To support reproducibility, we provide implementation details, hyperparameters, verifier architecture, representation extraction, dataset statistics, and additional experimental settings in Appendix B.

REFERENCES

William Abend, Emilio Bizzi, and Pietro Morasso. Human arm trajectory formation. *Brain: a journal of neurology*, 1982.

Zena Al-Khalili, Rafi Hakim, Dietrich Klakow, and Ji-Ung Lee. Fork-think with confidence. In *COLM*, 2026.

Jose A Arjona-Medina, Michael Gillhofer, Michael Widrich, Thomas Unterthiner, Johannes Brandstetter, and Sepp Hochreiter. Rudder: Return decomposition for delayed rewards. In *NeurIPS*, 2019.

Maria Attarian, Ian Vyse, Claas Voelcker, Jasper Gerigk, Evgenii Opryshko, Anas Almasri, Sumeet Singh, Yilun Du, and Igor Gilitschenski. Update-free on-policy steering via verifiers. *arXiv preprint arXiv:2603.10282*, 2026.

Jose Barreiros, Andrew Beaulieu, Aditya Bhat, Rick Cory, Eric Cousineau, Hongkai Dai, Ching-Hsin Fang, Kunimatsu Hashimoto, Muhammad Zubair Irshad, Masha Itkina, et al. A careful examination of large behavior models for multitask dexterous manipulation. *arXiv preprint arXiv:2507.05331*, 2025.

Eric Bigelow, Ari Holtzman, Hidenori Tanaka, and Tomer Ullman. Forking paths in neural text generation. In *ICLR*, 2025.

Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.

Jun Cen, Siteng Huang, Yuqian Yuan, Kehan Li, Hangjie Yuan, Chaohui Yu, Yuming Jiang, Jiayan Guo, Xin Li, Hao Luo, et al. Rynnvla-002: A unified vision-language-action and world model. *arXiv preprint arXiv:2511.17502*, 2025.

Hyeonbeom Choi, Daechul Ahn, Youhan Lee, Taewook Kang, Seongwon Cho, and Jonghyun Choi. Scale: Self-uncertainty conditioned adaptive looking and execution for vision-language-action models. In *ICML*, 2026.

Mingtong Dai, Lingbo Liu, Yongjie Bai, Yang Liu, Zhouxia Wang, Rui Su, Chunjie Chen, Liang Lin, and Xinyu Wu. Rover: Robot reward model as test-time verifier for vision-language-action model. *arXiv preprint arXiv:2510.10975*, 2025.

Jiafei Duan, Wilbert Pumacay, Nishanth Kumar, Yi Ru Wang, Shulin Tian, Wentao Yuan, Ranjay Krishna, Dieter Fox, Ajay Mandlekar, and Yijie Guo. Aha: A vision-language-model for detecting and reasoning over failures in robotic manipulation. In *ICLR*, 2025.

Shenyuan Gao, William Liang, Kaiyuan Zheng, Ayaan Malik, Seonghyeon Ye, Sihyun Yu, Wei-Cheng Tseng, Yuzhu Dong, Kaichun Mo, Chen-Hsuan Lin, et al. Dreamdojo: A generalist robot world model from large-scale human videos. In *ICML*, 2026.

Qiao Gu, Yuanliang Ju, Shengxiang Sun, Igor Gilitschenski, Haruki Nishimura, Masha Itkina, and Florian Shkurti. Safe: Multitask failure detection for vision-language-action models. In *NeurIPS*, 2025.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *ICLR*, 2022.

Physical Intelligence, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, et al. $\pi^*_{0.6}$: a vla that learns from experience. *arXiv preprint arXiv:2511.14759*, 2025a.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025b.

Suhyeok Jang, Dongyoung Kim, Changyeon Kim, Youngsuk Kim, and Jinwoo Shin. Verifier-free test-time sampling for vision language action models. In *ICLR*, 2026.

Zhewei Kang, Xuandong Zhao, and Dawn Song. Scalable best-of-n selection for large language models via self-certainty. In *NeurIPS*, 2025.

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. In *CoRL*, 2024.

Jacky Kwok, Christopher Agia, Rohan Sinha, Matt Foutter, Shulu Li, Ion Stoica, Azalia Mirhoseini, and Marco Pavone. Robomonkey: Scaling test-time sampling and verification for vision-language-action models. In *CoRL*, 2025.

Jacky Kwok, Shulu Li, Pranav Atreya, Yuejiang Liu, Yixing Jiang, Chelsea Finn, Marco Pavone, Ion Stoica, and Azalia Mirhoseini. Llm-as-a-verifier: A general-purpose verification framework. *arXiv preprint arXiv:2607.05391*, 2026a.

Jacky Kwok, Xilun Zhang, Mengdi Xu, Yuejiang Liu, Azalia Mirhoseini, Chelsea Finn, and Marco Pavone. Scaling verification can be more effective than scaling policy learning for vision-language-action alignment. *arXiv preprint arXiv:2602.12281*, 2026b.

Chengmeng Li, Junjie Wen, Yaxin Peng, Yan Peng, and Yichen Zhu. Pointvla: Injecting the 3d world into vision-language-action models. In *RA-L*, 2026.

Anthony Liang, Yigit Korkmaz, Jiahui Zhang, Minyoung Hwang, Abrar Anwar, Sidhant Kaushik, Aditya Shah, Alex S Huang, Luke Zettlemoyer, Dieter Fox, et al. Robometer: Scaling general-purpose robotic reward models via trajectory comparisons. *arXiv preprint arXiv:2603.02115*, 2026.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *ICLR*, 2024.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. In *NeurIPS*, 2023.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

Guanxing Lu, Wenkai Guo, Chubin Zhang, Yuheng Zhou, Haonan Jiang, Zifeng Gao, Yansong Tang, and Ziwei Wang. Vla-rl: Towards masterful and general robotic manipulation with scalable reinforcement learning. *arXiv preprint arXiv:2505.18719*, 2025.

Yueen Ma, Zixing Song, Yuzheng Zhuang, Jianye Hao, and Irwin King. A survey on vision–language–action models for embodied ai. *IEEE Transactions on Neural Networks and Learning Systems*, 2026.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. In *NeurIPS*, 2023.

Pietro Morasso. Spatial control of arm movements. *Experimental brain research*, 1981.

Soroush Nasiriany, Sepehr Nasiriany, Abhiram Maddukuri, and Yuke Zhu. Robocasa365: A large-scale simulation framework for training and benchmarking generalist robots. In *ICLR*, 2026.

Koichi Ogawara, Jun Takamatsu, Hiroshi Kimura, and Katsushi Ikeuchi. Extraction of essential interactions through multiple observations of human demonstrations. *IEEE Transactions on Industrial Electronics*, 2003.

Seongheon Park, Wendi Li, Changdae Oh, Samuel Yeh, Zsolt Kira, Michael Hagenow, and Sharon Li. Hide-and-seek in trajectories: Discovering failure signals for vla runtime monitoring. *arXiv preprint arXiv:2605.30834*, 2026.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

Delin Qu, Haoming Song, Qizhi Chen, Yuanqi Yao, Xinyi Ye, Yan Ding, Zhigang Wang, JiaYuan Gu, Bin Zhao, Dong Wang, et al. Spatialvla: Exploring spatial representations for visual-language-action model. In *RSS*, 2025.

Andrea Rosasco, Federico Ceola, Giulia Pasquale, and Lorenzo Natale. Kdpe: A kernel density estimation strategy for diffusion policy trajectory selection. In *CoRL*, 2025.

Hao Shi, Bin Xie, Yingfei Liu, Lin Sun, Fengrong Liu, Tiancai Wang, Erjin Zhou, Haoqiang Fan, Xiangyu Zhang, and Gao Huang. Memoryvla: Perceptual-cognitive memory in vision-language-action models for robotic manipulation. In *ICLR*, 2026.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Ajay Sridhar, Jennifer Pan, Satvik Sharma, and Chelsea Finn. Memer: Scaling up memory for robot control via experience retrieval. In *ICLR*, 2026.

Emanuel Todorov and Zoubin Ghahramani. Analysis of the synergies underlying complex hand manipulation. In *The 26th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE, 2004.

Marcel Torne, Karl Pertsch, Homer Walke, Kyle Vedder, Suraj Nair, Brian Ichter, Allen Z Ren, Haohuan Wang, Jiaming Tang, Kyle Stachowicz, et al. Mem: Multi-scale embodied memory for vision language action models. *arXiv preprint arXiv:2603.03596*, 2026.

Raymond Tsao, Andrew Wagenmaker, and Sergey Levine. Learning process rewards via success visitation matching for efficient rl. *arXiv preprint arXiv:2606.23640*, 2026.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *ACL*, 2024.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xiong-Hui Chen, Jianxin Yang, Zhenru Zhang, et al. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning. In *NeurIPS*, 2025.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *ICLR*, 2023.

A Michael West Jr and Neville Hogan. Kinematic hand synergies differ between reach-and-grasp and functional object manipulation. *Journal of Neurophysiology*, 2025.

Siyuan Yang, Yang Zhang, Haoran He, Ling Pan, Xiu Li, Chenjia Bai, and Xuelong Li. Steering vision-language-action models as anti-exploration: A test-time scaling approach. *arXiv preprint arXiv:2512.02834*, 2025.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, et al. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026.

Jianshu Zhang, Keliang Wu, Haoran Lu, Anbang Liu, Ce Zhang, Weijie Yin, Chengxuan Qian, Xiyuan Yang, Zhenyu Pan, Guo Ye, et al. Progress reward modeling for robotic learning: A comprehensive survey. *arXiv preprint arXiv:2607.21655*, 2026.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. The lessons of developing process reward models in mathematical reasoning. In *ACL Findings*, 2025.

Appendix

CONTENTS

A Algorithm **15**

B Experimental Details **15**

B.1 Implementation 15

B.2 Hyperparameters 16

B.3 Dataset statistics 16

B.4 Evaluation Tasks 16

C Additional Analysis **17**

C.1 Does Measured Criticality Reflect Actual Decision Consequence? 17

C.2 Autoregressive Model 18

C.3 Inference Latency of Action Sampling 18

C.4 Effect of the Number of Samples K for Criticality Estimation 18

C.5 Representation Extraction 19

C.6 Visualization of Action-Expert Representations 19

D Derivation of the Success-Visitation Ratio **20**

E Baselines **21**

F Additional Literature Survey **22**

G Limitations and Future Work **22**

A ALGORITHM

Algorithm 1 Decision-weighted verifier learning and Best-of-N inference

Require: Frozen VLA policy π_{pre}, rollout trajectories $\mathcal{T}$, criticality samples K, test-time candidates N, training epochs E, weighting parameter β

1. Decision-Weighted Dataset Construction

$\mathcal{D} \leftarrow \emptyset$
for each trajectory $\tau \in \mathcal{T}$ with outcome y_τ **do**
  for $t = 1, \ldots, T_\tau$ **do**
    Extract the executed-action representation z_t
    Sample K candidate action chunks $\{\tilde{\mathbf{a}}_t^{(k)}\}_{k=1}^{K} \sim \pi_{\text{pre}}(\cdot \mid s_t)$
    Extract candidate representations $\{z_t^{(k)}\}_{k=1}^{K}$
    Compute criticality score u_t using Equation (8)
  end for
  Compute trajectory statistics μ_τ and σ_τ over $\{u_t\}_{t=1}^{T_\tau}$
  for $t = 1, \ldots, T_\tau$ **do**
    Compute standardized criticality $\tilde{u}_t$ and weight w_t using Equation (9)
    Set $y_t \leftarrow y_\tau$
    Add (z_t, y_t, w_t) to $\mathcal{D}$
  end for
end for

2. Verifier Training

for $e = 1, \ldots, E$ **do**
  for each class-balanced minibatch $\mathcal{B} \subset \mathcal{D}$ **do**
    Update θ using the decision-weighted BCE in Equation (11)
  end for
end for

3. Verifier-Guided Best-of-N Inference

for each decision step t **do**
  Sample N candidate action chunks $\{\mathbf{a}_t^{(i)}\}_{i=1}^{N} \sim \pi_{\text{pre}}(\cdot \mid s_t)$
  Extract action-expert representations $\{z_t^{(i)}\}_{i=1}^{N}$
  Score each candidate using the verifier logit $\widehat{f}_\theta(z_t^{(i)})$
  Select and execute the highest-scoring candidate using Equation (2)
end for

B EXPERIMENTAL DETAILS

B.1 IMPLEMENTATION

Verifier architecture and representation extraction. We implement the verifier as a two-layer MLP operating on internal hidden representations from the VLA action expert. The MLP uses hidden dimensions of 64 with a dropout rate of 0.3, followed by a sigmoid output layer. We extract the final-layer action-token representations at the final denoising step and average them across action tokens. The resulting representation is used for both decision-criticality estimation and verifier training. Ablations on the representation layer and token position are provided in Section C.5.

Training. We train the verifier for 30 epochs using AdamW (Loshchilov & Hutter, 2019) with a learning rate of 10^{-3}, weight decay of 10^{-4}, batch size 512, and class-balanced sampling. For decision-weighted training, we set $\beta = 0.5$ and clip w_t to $[c^{-1}, c]$ with $c = 2.5$ to limit the influence of extreme criticality values. We sample $K = 32$ action candidates at each training state to estimate decision criticality from action-representation variance. The VLA policy remains frozen throughout training.

Software and hardware. Our implementation uses PyTorch (Paszke et al., 2019). All training and inference experiments are conducted on a single NVIDIA RTX A6000 GPU with 48 GB of memory.

B.2 HYPERPARAMETERS

Table 7: Hyperparameter search space applied uniformly across all experiments. Selected values are **bolded**.

Hyperparameter	Search Space
Optimization	
Optimizer	{Adam, **AdamW**, SGD}
Learning rate	{1e-2, 5e-3, 1e-3, **5e-4**, 1e-4}
Training epochs	{10, **30**, 50, 100}
Architecture	
MLP hidden size	{**64**, 256, 512}
Decision weighting	
Weight temperature β	{0.1, 0.25, **0.5**, 1.0, 2.0}
Weight clipping threshold c	{1.5, 2.0, **2.5**, 3.0, 5.0}

We tune the verifier hyperparameters on RoboCasa using $\pi_{0.5}$ and then fix the selected configuration for all remaining benchmarks and VLA policies. This protocol avoids benchmark-specific tuning and evaluates whether a single hyperparameter configuration transfers across different environments and policy backbones.

B.3 DATASET STATISTICS

Table 8: **Dataset statistics.** Number of successful and failed episodes and their corresponding environment steps in the offline rollout datasets collected with (a) π_0 and (b) $\pi_{0.5}$.

(a) π_0

	Success		Failure	
Dataset	# Episodes	# Steps	# Episodes	# Steps
LIBERO	249	66,320	51	26,520
RoboCasa	184	60,285	356	225,600
Real Robot	-	-	-	-

(b) $\pi_{0.5}$

	Success		Failure	
Dataset	# Episodes	# Steps	# Episodes	# Steps
LIBERO	281	73,140	19	9,880
RoboCasa	248	92,985	292	120,900
Real Robot	94	12,387	66	12,012

Our verifier training data are collected directly from rollouts of the VLA policy, where both successful and failed episodes arise naturally from policy execution. As shown in Table 8, these datasets exhibit class imbalance across policies and environments. Despite this imbalance, our verifier can be trained effectively without artificially balancing the outcome distribution during data collection.

B.4 EVALUATION TASKS

We evaluate our method across three settings: the LIBERO-Long benchmark, the RoboCasa atomic-task benchmark, and a real-world manipulation setup using a Franka Research 3 robot. The complete task lists are provided in Tables 9 to 11.

Table 9: **Real-world robot evaluation tasks.** We evaluate on four manipulation tasks spanning object placement, stacking, and articulated-object interaction.

ID	Task	Skill
1	Place the green block in the bowl	Pick & Place
2	Place the cup on the plate	Pick & Place
3	Stack the red block on the blue block	Stacking
4	Close the drawer	Articulated Manipulation

Table 10: **Tasks in the LIBERO-Long benchmark.** We evaluate on all 10 long-horizon manipulation tasks in the standard LIBERO-Long suite.

ID	Task
1	Put both the alphabet soup and the tomato sauce in the basket
2	Put both the cream cheese box and the butter in the basket
3	Turn on the stove and put the moka pot on it
4	Put the black bowl in the bottom drawer of the cabinet and close it
5	Put the white mug on the left plate and put the yellow and white mug on the right plate
6	Pick up the book and place it in the back compartment of the caddy
7	Put the white mug on the plate and put the chocolate pudding to the right of the plate
8	Put both the alphabet soup and the cream cheese box in the basket
9	Put both moka pots on the stove
10	Put the yellow and white mug in the microwave and close it

Table 11: **RoboCasa evaluation tasks.** The 18 Atomic-Seen tasks span four broad interaction categories: Articulated (7), Control (4), Object-Centric (6), and Navigation (1). These categories consolidate the finer-grained atomic skills defined by RoboCasa365.

Category	Task	RoboCasa Skill
Articulated	Close blender lid	Open/Close Lid
	Close fridge	Open/Close Door
	Close toaster oven door	Open/Close Door
	Open cabinet	Open/Close Door
	Open drawer	Open/Close Drawer
	Open stand mixer head	Open/Close Lid
	Slide dishwasher rack	Slide Rack
Control	Turn off stove	Twist Knob
	Turn on sink faucet	Turn Lever
	Turn on electric kettle	Press Button
	Turn on microwave	Press Button
Object-Centric	Coffee setup mug	Insertion
	Pick place: counter → cabinet	Pick & Place
	Pick place: counter → stove	Pick & Place
	Pick place: drawer → counter	Pick & Place
	Pick place: sink → counter	Pick & Place
	Pick place: toaster → counter	Pick & Place
Navigation	Navigate kitchen	Navigation

C Additional Analysis

C.1 Does Measured Criticality Reflect Actual Decision Consequence?

Our method uses action-representation variance as a rollout-free proxy for decision criticality, based on the intuition that candidate selection should matter more when plausible actions diverge. We directly test this assumption by comparing u_t with rollout-based estimates of the downstream consequence of selecting different action candidates.

We randomly sample 500 states from RoboCasa using $\pi_{0.5}$. For each saved simulator state s_t, we sample K action candidates $\{a_t^{(k)}\}_{k=1}^{K}$ and compute the corresponding representation-based criticality u_t. To measure the downstream consequence of each candidate, we branch the simulator from the same state, execute $a_t^{(k)}$,

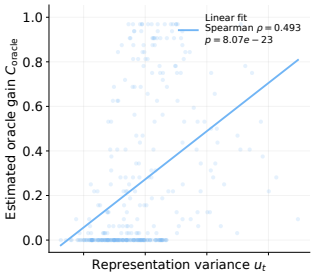


Figure 7: Measured criticality u_t vs. oracle criticality C_{oracle}.

and roll out the frozen base policy to estimate its success probability (Wang et al., 2024),

$$Q(s_t, a_t^{(k)}) = P(\text{success} \mid s_t, a_t^{(k)}, \pi_{0.5}).$$

We define rollout-based oracle criticality as the potential gain from selecting the best candidate rather than choosing uniformly at random:

$$C_{\text{oracle}}(s_t) = \max_k Q(s_t, a_t^{(k)}) - \frac{1}{K}\sum_{k=1}^{K} Q(s_t, a_t^{(k)}).$$

This quantity is small when candidate selection has little influence on the final outcome and large when selecting the appropriate action is consequential. As shown in Figure 7, our measured criticality u_t is positively correlated with rollout-based oracle criticality (Spearman $\rho = 0.493$, $p = 8.09 \times 10^{-23}$). Together with the gains from criticality-weighted verifier learning over uniform training in Tables 3 and 4 and the stronger downstream performance of action-representation features over raw actions in Table 5, these results provide complementary evidence that action-representation variance identifies states where candidate selection has meaningful downstream consequences and provides an effective signal for verifier learning.

C.2 Autoregressive Model

While our main experiments focus on flow-matching VLAs, we further evaluate DiVeR with the autoregressive OpenVLA model to assess its applicability across different action-generation paradigms. As shown in Table 12, with $N = 4$ candidates, DiVeR improves the success rate by +3.0 percentage points over the pretrained policy, while outperforming MGSelect and SVM by +1.0 and +0.5 percentage points, respectively. These results indicate that our verifier-guided selection framework is not specific to flow-matching policies and can also be effectively applied to autoregressive VLAs.

Table 12: Success rate (%) with OpenVLA using $N = 4$ candidates on LIBERO-Long.

Method	Success (%)
Base	52.5
MGS	54.5
SVM	55.0
Ours	**55.5**

C.3 Inference Latency of Action Sampling

We measure the chunk-level inference latency required to generate multiple action candidates from the frozen VLA policy. As shown in Figure 8, sampling latency increases with the number of candidates N, reaching approximately 360 ms for π_0 and 205 ms for $\pi_{0.5}$ at $N = 16$. We use $N = 16$ as the default inference setting, providing a practical trade-off between candidate diversity and sampling cost. In comparison, our lightweight verifier requires *less than* 0.5 *ms* to score each action chunk, adding minimal overhead relative to VLA action generation. Thus, the inference cost of DiVeR is dominated by candidate sampling rather than verifier evaluation.

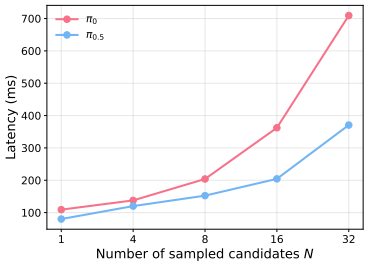


Figure 8: Chunk-level action sampling latency.

C.4 Effect of the Number of Samples K for Criticality Estimation

We study the effect of the number of action samples K used to estimate state criticality on RoboCasa with $\pi_{0.5}$. As shown in Table 13, performance improves as K increases from 4 to 32, indicating that a larger candidate set provides a more reliable estimate of action disagreement. Increasing K further to 64 yields only marginal improvement while requiring additional sampling cost. We therefore use $K = 32$ throughout our experiments, which provides a favorable trade-off between the quality of criticality estimation quality and computational cost during data collection.

Table 13: Effect of the number of samples K used for criticality estimation.

K	Success (%)
4	52.8
16	53.9
32	54.7
64	55.0

C.5 Representation Extraction

Table 14: **Representation extraction ablation on RoboCasa with $\pi_{0.5}$.** Success rate (%) under different (a) action-token positions and (b) action-expert layers.

(a) Token position

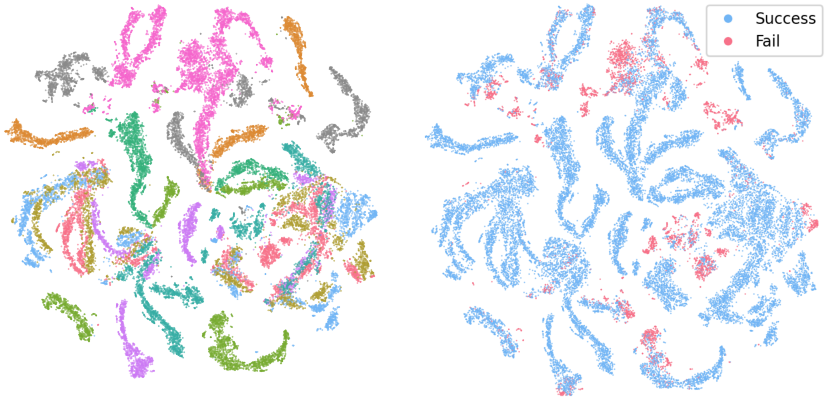
Position	Success Rate (%)
First	53.6
Last	51.5
Average	**54.7**

(b) Action-expert layer

Layer	Success Rate (%)
First	52.9
Middle	44.1
Last	**54.7**

We study how the choice of action-token position and action-expert layer affects verifier performance. This ablation is conducted on RoboCasa using $\pi_{0.5}$. We compare representations extracted from the first, last, or averaged action-token positions, as well as from the first, middle, or final action-expert layers. As shown in Table 14, averaging the action-token representations yields the highest success rate among the token-position variants, while the final action-expert layer performs best among the layer choices. We therefore use the average action-token representation from the final action-expert layer as our default representation for both decision-criticality estimation and verifier training.

C.6 Visualization of Action-Expert Representations



(a) Task-wise structure (b) Success vs. failure

Figure 9: **t-SNE visualization of action-expert representations.** We visualize action-expert representations from a frozen π_0 policy over 500 LIBERO-Long episodes. (a) Different colors denote different tasks, revealing task-dependent structure in the representation space. (b) Successful and failed trajectories occupy distinct regions of the representation space.

We further analyze the internal action-expert representations used by the verifier. As shown in Figure 9(a), representations exhibit structured task dependent organization, with samples from the same task tending to occupy similar regions of the embedding space. This suggests that the representation retains task related context that can support a single verifier shared across multiple tasks.

We further examine whether the representation contains information related to behavioral outcomes. For this analysis, representations after failure onset are labeled as failure representations. As shown in Figure 9(b), successful and failed behaviors exhibit partially distinct patterns, with failure representations concentrated in particular regions of the embedding space. This suggests that the internal representation contains outcome related information that can provide useful features for verifier learning.

Together, these visualizations suggest that the action-expert representation contains both task related structure and outcome related variation. While the visualization alone does not establish that the

outcome related structure is fully shared across unseen tasks, it is consistent with the cross task generalization observed in Figure 5.

D Derivation of the Success-Visitation Ratio

We provide the derivation showing that the logit of a binary discriminator trained to distinguish successful from failed state–action visitation recovers their log density ratio from Section 4.1. Let $\rho^+(s,a)$ and $\rho^-(s,a)$ denote the normalized state–action visitation densities induced by successful and failed trajectories, respectively. Assuming balanced sampling between the two classes, we train a discriminator $f(s,a) \in (0,1)$ with the binary cross-entropy objective

$$\mathcal{L}(f) = -\mathbb{E}_{(s,a)\sim\rho^+}[\log f(s,a)] - \mathbb{E}_{(s,a)\sim\rho^-}[\log(1-f(s,a))]. \tag{12}$$

Proposition 1. *The population-optimal discriminator satisfies*

$$f^\star(s,a) = \frac{\rho^+(s,a)}{\rho^+(s,a)+\rho^-(s,a)}. \tag{13}$$

Consequently, its logit recovers the log success-to-failure visitation density ratio:

$$\log \frac{f^\star(s,a)}{1-f^\star(s,a)} = \log \frac{\rho^+(s,a)}{\rho^-(s,a)}. \tag{14}$$

Proof. Because the objective decomposes pointwise over state–action pairs, we can minimize the contribution of each (s,a) independently. For a fixed (s,a), let

$$p = \rho^+(s,a), \qquad q = \rho^-(s,a),$$

and write $f = f(s,a)$. The corresponding pointwise loss is

$$\ell(f) = -p\log f - q\log(1-f). \tag{15}$$

Differentiating with respect to f gives

$$\frac{\partial \ell}{\partial f} = -\frac{p}{f} + \frac{q}{1-f}. \tag{16}$$

Setting the derivative to zero,

$$-\frac{p}{f} + \frac{q}{1-f} = 0, \tag{17}$$

which implies

$$qf = p(1-f). \tag{18}$$

Therefore,

$$f^\star = \frac{p}{p+q} = \frac{\rho^+(s,a)}{\rho^+(s,a)+\rho^-(s,a)}. \tag{19}$$

Moreover,

$$\frac{\partial^2 \ell}{\partial f^2} = \frac{p}{f^2} + \frac{q}{(1-f)^2} > 0, \tag{20}$$

so this stationary point is the unique minimizer whenever $p+q>0$.

Finally,

$$\frac{f^\star(s,a)}{1-f^\star(s,a)} = \frac{\rho^+(s,a)/(\rho^+(s,a)+\rho^-(s,a))}{\rho^-(s,a)/(\rho^+(s,a)+\rho^-(s,a))} \tag{21}$$

$$= \frac{\rho^+(s,a)}{\rho^-(s,a)}. \tag{22}$$

Taking the logarithm of both sides yields

$$\log \frac{f^\star(s,a)}{1-f^\star(s,a)} = \log \frac{\rho^+(s,a)}{\rho^-(s,a)}, \tag{23}$$

which proves the result. □

Effect of decision weighting. The proposition above characterizes the unweighted BCE objective and provides the density-ratio interpretation of our base verifier. DIVER further reweights this objective according to decision criticality, placing greater learning emphasis on states where accurate candidate discrimination is more consequential. Accordingly, we interpret the weighted verifier score as a candidate-ranking score rather than an exact density-ratio estimate.

E BASELINES

All selection rules operate under an identical Best-of-N protocol. At each decision step t, we draw a single candidate set $\{\mathbf{a}_t^{(i)}\}_{i=1}^N \sim \pi_{\text{pre}}(\cdot \mid s_t)$ from the same frozen policy and share it across all methods, so that differences in performance are attributable solely to the selection rule rather than to the proposal distribution. Each method defines a scalar score $g(s_t, \mathbf{a}_t^{(i)}) \in \mathbb{R}$, and the robot executes $\mathbf{a}_t^\star = \arg\max_{i\in[N]} g(s_t, \mathbf{a}_t^{(i)})$ as in Equation (2). We write $\mathbf{a} \in \mathbb{R}^{H\cdot d_a}$ for the flattened, normalized action chunk, where H is the action horizon and d_a is the per-step action dimension, and $z \in \mathbb{R}^d$ for the action-expert representation defined in Section 4. Learned baselines are trained on the same offline rollout dataset, with the same optimizer, batch size, and class-balanced sampling as our verifier.

KDPE (Rosasco et al., 2025). KDPE is verifier-free and selects the candidate that best represents the policy's own action distribution, discarding low-density (potentially out-of-distribution) samples. Using the N candidates as the sample set, it forms a leave-one-out kernel density estimate in action-chunk space,

$$\widehat{p}_h(\mathbf{a} \mid s_t) = \frac{1}{N-1}\sum_{j\neq i} K_h\left(\mathbf{a}, \mathbf{a}_t^{(j)}\right), \qquad K_h(x,y) \propto \exp\left(-\frac{\|x-y\|_2^2}{2h^2}\right), \tag{24}$$

and scores each candidate by its estimated density, $g_{\text{KDPE}}(s_t, \mathbf{a}_t^{(i)}) = \widehat{p}_h(\mathbf{a}_t^{(i)} \mid s_t)$. The bandwidth is set per state by Silverman's rule, $h = \widehat{\sigma}\, N^{-1/(d_{\text{eff}}+4)}$, where $\widehat{\sigma}$ is the mean per-coordinate standard deviation of the candidate set. The original formulation defines the kernel on the $SE(3)$ manifold of end-effector poses; because our policies emit joint-space action chunks, we apply an isotropic Gaussian kernel in the normalized action space. This rule is a mode-seeking heuristic: it favors the most typical candidate and is therefore risk-averse, but it is uninformed by task outcome and cannot prefer a rare candidate even when the majority mode leads to failure.

MG-Select (Jang et al., 2026). MG-Select is also verifier-free and ranks candidates by how strongly the conditioning information supports them, in the spirit of classifier-free guidance. In its original autoregressive instantiation the score is the log-likelihood gap between the conditioned and condition-masked policy, $\log \pi_{\text{pre}}(\mathbf{a} \mid s_t) - \log \pi_{\text{pre}}(\mathbf{a} \mid \tilde{s}_t)$, where $\tilde{s}_t$ denotes the observation with the conditioning signal masked out. Our flow-matching policies expose no token likelihoods, so we realize this quantity as a kernel-density ratio. We draw an auxiliary reference set of $M = N$ chunks from the masked-conditioning policy, $\tilde{\mathbf{a}}_t^{(m)} \sim \pi_{\text{pre}}(\cdot \mid \tilde{s}_t)$, estimate both densities with the kernel of Equation (24), and score

$$g_{\text{MGS}}(s_t, \mathbf{a}_t^{(i)}) = \log \widehat{p}_h\left(\mathbf{a}_t^{(i)} \mid s_t\right) - \log \widehat{p}_h\left(\mathbf{a}_t^{(i)} \mid \tilde{s}_t\right), \tag{25}$$

which is a sample-based estimate of the pointwise mutual information between the candidate and the conditioning. The selected chunk is the one whose trajectory-space density is most amplified by conditioning. We use the same kernel and bandwidth rule as KDPE so that the two verifier-free baselines differ only in the use of the masked reference set.

TACO (Yang et al., 2025). TACO casts test-time selection as anti-exploration: rather than seeking novelty, it prefers candidates with high pseudo-count under the offline data distribution. Following the original formulation, we instantiate the pseudo-count with random network distillation. A target network $\bar{f} : \mathbb{R}^d \to \mathbb{R}^m$ is randomly initialized and frozen, and a predictor f_ψ is fit on the rollout dataset by

$$\min_\psi \ \mathbb{E}_{(s,\mathbf{a})\sim\mathcal{D}} \left\| f_\psi(z) - \bar{f}(z) \right\|_2^2, \tag{26}$$

so that the residual is small on in-distribution state–action pairs and large elsewhere. Candidates are scored by the negative prediction error, $g_{\mathrm{TACO}}(s_t, \mathbf{a}_t^{(i)}) = -\|f_\psi(z_t^{(i)}) - \bar{f}(z_t^{(i)})\|_2^2$. We fit f_ψ on the same action-expert representations used by our verifier, which removes the representation space as a confounder. The key distinction from our method is supervision: TACO measures support under the behavior distribution and never observes trajectory outcomes, so it cannot distinguish an in-distribution action that reliably fails from one that succeeds.

SVM (Tsao et al., 2026). Success Visitation Matching Verifier (SVM) is the learned verifier most closely related to ours: it discriminates successful from failed state–action visitation using trajectory-level outcome labels. Let ρ^+ and ρ^- denote the visitation densities induced by successful and failed trajectories. A discriminator $D_\phi(s, \mathbf{a}) \in (0, 1)$ is trained with unweighted binary cross-entropy,

$$\mathcal{L}_{\mathrm{SVM}}(\phi) = -\mathbb{E}_{(s,\mathbf{a})\sim\rho^+}\left[\log D_\phi(s, \mathbf{a})\right] - \mathbb{E}_{(s,\mathbf{a})\sim\rho^-}\left[\log(1 - D_\phi(s, \mathbf{a}))\right], \tag{27}$$

and its logit is used as the score, $g_{\mathrm{SVM}}(s_t, \mathbf{a}_t^{(i)}) = \log \frac{D_\phi}{1-D_\phi}$, which estimates the log success-visitation ratio $\log \rho^+/\rho^-$. Following the original implementation, we encode the current image observation with a CNN, concatenate the resulting feature with the corresponding raw action chunk, and train an MLP discriminator on the concatenated input. SVM therefore differs from DIVER along exactly two axes: (i) the input space (image features with raw actions, rather than the policy's action-expert representation), and (ii) the weighting of training samples (uniform over all visited timesteps, rather than reweighted by decision criticality). Table 4 isolates these two factors.

F Additional Literature Survey

Structured variability in human motor control. Human motor variability changes systematically with task structure rather than remaining uniform throughout movement. Simple reaching exhibits highly stereotyped trajectories (Morasso, 1981; Abend et al., 1982), while complex object manipulation recruits a higher-dimensional repertoire of hand coordination patterns than simple grasping (Todorov & Ghahramani, 2004; West Jr & Hogan, 2025). Related robotics work further shows that structured variation across repeated human demonstrations can reveal task-relevant interaction points (Ogawara et al., 2003). Together, these findings support our view that different parts of a trajectory are not equally informative, motivating greater emphasis on decision-critical states during verifier learning.

Non-uniform supervision and temporal credit assignment. Related work in reward modeling, failure detection, and sequential decision making highlights the value of learning signals that distinguish individual steps within a trajectory. Process reward models provide feedback on intermediate steps rather than only final outcomes (Lightman et al., 2024; Wang et al., 2024; Lu et al., 2025), while trajectory-supervised failure detection identifies temporally localized failure signals in VLA execution without step-level annotations (Park et al., 2026). Similarly, Arjona-Medina et al. (2019) address temporal credit assignment by redistributing delayed rewards to earlier state–action pairs according to their estimated contributions to the return. These approaches motivate moving beyond uniform supervision across a trajectory and emphasizing states where action selection has greater consequences for task success.

G Limitations and Future Work

Our study has several limitations. First, although we evaluate DiVeR in both simulation and on a real Franka Research 3 robot, the real-world experiments cover a limited set of manipulation tasks and do not fully characterize transfer across broader environments or robot embodiments. Second, DiVeR selects among candidates proposed by a frozen policy, so its performance ultimately depends on the quality and diversity of the available action candidates. Future work could explore broader real-world evaluation and stronger candidate generation mechanisms.