Looking Back to Move Forward: Temporal Verification for Generative Robot Policies

Anonymous Author(s)
Affiliation
Address
email

Abstract

Generative policies have emerged as a promising paradigm for robot learning, combining expressive generative action modeling with scalable imitation learning from large demonstration corpora. Yet when trained on heterogeneous demonstrations, their chunked generation process can produce suboptimal actions, causing errors to compound during execution and eventually driving the robot into out-of-distribution failure states. Action verification has recently emerged as a test-time scaling approach to mitigate this issue, sampling multiple action candidates and using an external verifier to select the best one for execution. However, existing approaches remain temporally myopic and impractical in design: they evaluate candidates using only static-timestep observations and often rely on large-scale verifiers and additional expert demonstrations. In this paper, we introduce **Te**mporal **V**erification (**TeV**), an efficient temporally aware action verification framework for flow-matching VLAs. TeV first learns a *temporal token* that summarizes recent observation-action history, allowing candidate chunks to be evaluated as continuations of the robot's execution trajectory rather than isolated predictions. Conditioned on this token, TeV constructs positive–negative sample pairs without additional expert data or preference annotations, and trains an energy-based verifier with a contrastive objective to favor higher-quality chunks that are temporally compatible with recent execution. Beyond post-hoc ranking, TeV uses the learned energy score to steer intermediate samples toward lower-energy regions, potentially improving the quality of generated candidates. Extensive experiments in simulation and real-world settings demonstrate that TeV provides reliable action candidate ranking, improves task success, and yields smoother execution trajectories.

1 Introduction

Generative policies based on diffusion and flow models have rapidly become a central paradigm in robot learning [8, 23, 2]. By scaling imitation learning to large collections of human demonstrations, these models have enabled robots to perform increasingly complex skills, including household cleaning [16], cloth folding [47], and contact-rich assembly such as screw driving [45]. In particular, Vision-Language-Action (VLA) policies trained with flow-matching objectives [23] have emerged as a promising foundation for generalist robot control [16, 32, 29, 42, 7]. These models couple the semantic reasoning capacity of vision-language backbones with continuous action generation, allowing language instructions, visual observations, and robot states to be mapped into temporally extended action chunks. Flow-matching further provides an efficient generative objective for action sampling, making it especially attractive for scalable robot policy learning.

Despite this promise, the generative flexibility that makes flow-matching policies expressive also exposes a critical weakness. When trained on heterogeneous demonstrations, such a generation pipeline can produce suboptimal actions. Small errors may compound over time, gradually driving

Submitted to 40th Conference on Neural Information Processing Systems (NeurIPS 2026). Do not distribute.

the robot away from the demonstrated state-action manifold and into out-of-distribution states that culminate in failure. Previous methods address this issue by retraining or fine-tuning the base policy through reinforcement learning [25, 44, 48] or representation regularization [49, 18]. While effective, these approaches can compromise the base policy's generalist prior and incur substantial computational cost. Verification-based test-time scaling [1, 46] therefore offers a practical alternative, which improves deployment-time action selection without modifying the base policy. Rather than committing to a single sampled action chunk, these methods allocate additional inference-time compute to sample multiple candidate chunks from the underlying policy. An external verifier is then used to select the candidate most aligned with the expert distribution or preference signal [27, 46].

Yet existing verification-based methods face two limitations. First, they suffer from temporal myopia, where candidate actions are generated and evaluated fully based on observations from the nearest static timestep. For chunked generative policies, successful execution depends not only on selecting high-quality candidates under the current observation, but also on ensuring that consecutive action chunks form a coherent trajectory. Without this cross-chunk consistency, the policy may oscillate between individually plausible modes, resulting in jerky motion, unstable execution, and task failure [24, 40]. Thus, effective verification must evaluate both *local action plausibility* and *temporal consistency* with the robot's recent execution history. The second limitation arise from the practical burden of their verifier choice. Existing approaches often rely on large VLM-based verifiers [19, 1], which require careful adaptation through prompt engineering and abundant task-specific supervision. Such supervision is expensive to obtain, since reward labels, trajectory preferences, and failure traces require additional expert annotation, costly rollout collection, or physical interaction [31, 22]. Large-scale verifiers also hinder real-world robot deployment because they are costly to run in latency-sensitive control loops [3, 32], especially under limited onboard compute [17].

To address these limitations, we propose **Te**mporal **V**erification (**TeV**), an efficient energy-based framework for flow-matching VLAs. TeV improves both the temporal consistency and quality of generated action chunks through two complementary functions: it evaluates completed candidates for execution and steers intermediate samples during test-time generation. Specifically, we first construct a *temporal token* that summarizes the robot's recent observation-action history. Produced by an efficient temporal encoder and regularized with a dynamics-aware objective, this token provides the verifier with execution context beyond the current observation. Conditioned on the temporal token, a lightweight energy model scores sampled action chunks and ranks them for candidate selection. Since verification only requires relative preference among candidates rather than calibrated reward estimation, we train the verifier contrastively: expert demonstration chunks serve as positives, while temporally mismatched chunks and policy-generated deviations serve as informative negatives. Finally, TeV extends verification beyond post-hoc ranking by using the differentiable energy score as guidance during flow integration, steering intermediate samples toward lower-energy regions that may yield better candidates before final selection.

Temporal Verification is a lightweight add-on module for flow-matching VLAs. Its verifier introduces less than 0.15% additional parameters relative to the base policy and requires no additional expert demonstrations. Extensive experiments in simulation and real-world settings show that TeV provides reliable energy-based ranking, yields consistent task-success gains of 6%–18%, and produces smoother robot dynamics, highlighting its effectiveness for generative robot control.

2 Related Work

Test-time Scaling via Action Verification. Test-time verification improves robot execution by allocating additional inference-time computation to evaluate and rank candidate outputs [33, 20, 19]. In robotics, this paradigm has been explored through several verifier designs. External-verifier methods use vision-language models to evaluate candidate actions or plans [19, 1], either with task-specific training or zero-shot visual reasoning. RL-based approaches support verification through online policy adaptation [22] or learned value functions for action ranking [27]. Other methods use verification to improve instruction following by checking whether generated actions remain aligned with language goals [20]. The most relevant verification approach to ours is TACO [46], which uses a pseudo-count estimator to favor candidate actions near high-density successful modes of the training distribution. However, existing methods rely on static timestep observations and may use abundant external supervision for action selection, whereas our method conditions verification on recent execution history to encourage temporally coherent and smooth behavior.

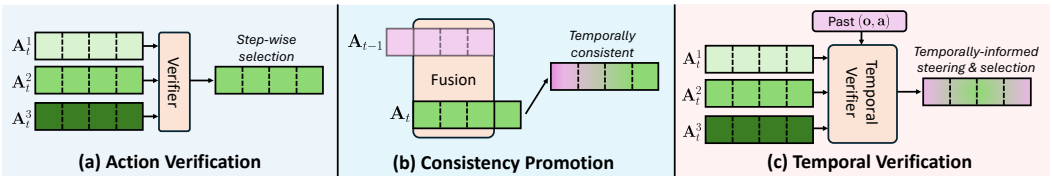

Figure 1: **Paradigm comparison.** **(a)** Conventional action verification selects among multiple sampled candidates at each decision step, relying only on the current observation. **(b)** Temporal-consistency methods refine the current prediction by using previous action chunks to promote cross-chunk coherence. **(c)** Temporal Verification integrates action verification with temporal consistency modeling. Conditioned on past observation-action context, the learned temporal verifier steers and selects action chunk candidates that are more consistent with recent execution.

Temporal Consistency for Chunked Policies. The chunked generation paradigm of diffusion and flow-matching policies can produce action chunks that are locally plausible but temporally inconsistent across consecutive predictions, leading to mode oscillation and jerky behaviors during deployment. Existing methods improve temporal consistency by modifying how action chunks are executed. ACT [50] smooths execution by averaging overlapping action chunks, while BID [24] samples multiple candidates and selects the chunk most consistent with the previous prediction over overlapping timesteps. Real-time control methods [3, 36, 40] study temporal consistency under asynchronous execution, where stale observations further complicate the problem. In contrast, our method learns a temporally aware verification signal from recent observation-action history, enabling an energy-based verifier to select candidates that are compatible with ongoing execution.

Policy Steering. Inference-time steering improves pretrained generative policies by modifying the sampling or refinement process while keeping the base policy fixed [27]. Existing methods typically use external objectives or auxiliary models to guide generated actions toward desired task, safety, or user-specified constraints. Model-predictive refinement methods [28, 34] guide policy outputs using learned dynamics models to satisfy safety constraints or improve task performance, while human-in-the-loop approaches [4, 43, 30] refine actions based on user-provided subgoals, corrections, or preferences. Classifier- and dynamics-guided methods [10, 35] use latent visual dynamics models [15, 41] to steer generated actions toward desired outcomes. Our method extends contrastive energy verification beyond post-hoc candidate ranking to enable inference-time steering. Figure 1 summarizes the relationship between our method and related works.

3 Preliminaries

Flow-matching-based VLAs. Flow-matching policies [23] generate actions by learning a continuous-time velocity field that transports samples from a simple prior, such as Gaussian noise, to the action distribution. In flow-matching VLAs, the current observation context is first encoded by the VLM backbone into a conditional representation $\mathcal{K}$. The action expert then conditions on $\mathcal{K}$ and transforms an initial noise sample $\mathbf{x}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ into an action chunk by integrating the learned velocity field $\mathbf{v}_\pi$ over normalized flow time $s \in [0, 1]$. With N Euler integration steps, the update is

$$\mathbf{x}_{s+\frac{1}{N}} = \mathbf{x}_s + \frac{1}{N}\mathbf{v}_\pi(\mathbf{x}_s, s \mid \mathcal{K}), \tag{1}$$

where integration starts from $\mathbf{x}_0$ and terminates at $\mathbf{x}_1$. The final output $\mathbf{x}_1 \in \mathbb{R}^{H\times D}$ is an action chunk of horizon H, with each action having dimension D.

Contrastive learning and energy-based models. Contrastive learning trains a scoring function to distinguish positive samples from negative ones according to relative compatibility. Energy-based models (EBMs) [21, 11] provide a natural formulation for such scoring: an input $\mathbf{z}$ is assigned a scalar energy $E_\theta(\mathbf{z}) \in \mathbb{R}$, where lower energy indicates higher compatibility. In conditional settings, the energy also depends on a context variable $\mathbf{c}$, written as $E_\theta(\mathbf{z} \mid \mathbf{c})$. Thus, a standard contrastive objective for EBMs is the margin-based ranking loss [13, 14]:

$$\mathcal{L}_{\text{margin}} = \left[m + E_\theta(\mathbf{z}^+ \mid \mathbf{c}) - E_\theta(\mathbf{z}^- \mid \mathbf{c})\right]_+, \tag{2}$$

where $\mathbf{z}^+$ and $\mathbf{z}^-$ denote positive and negative samples under the same context $\mathbf{c}$, $m > 0$ is a margin, and $[\cdot]_+ = \max(0, \cdot)$ is the hinge operator. The loss encourages positives to have lower energy

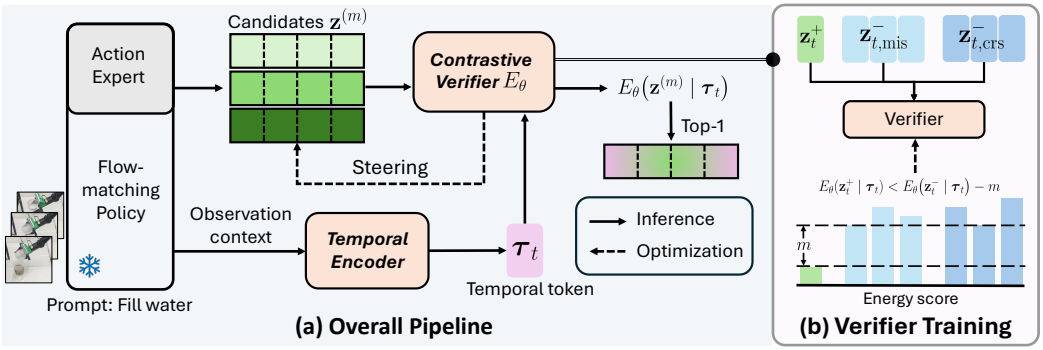

Figure 2: **Illustration of Temporal Verification. (a)** Overall pipeline. We encode observation-action history into a temporal token and use it to condition the energy-based contrastive verifier. At each flow-matching integration step, candidate representations are scored by the verifier. The resulting energy scores are used to steer intermediate samples during integration and rank completed chunks after integration. **(b)** Verifier training process. Expert chunks serve as positives, while temporally mismatched chunks and policy-generated deviations serve as negatives. The verifier is trained to assign positives lower energy than negatives by at least a specified margin.

than negatives by at least m. We observe that verification requires only relative ordering rather than calibrated absolute scores, which makes contrastive energy modeling well suited to verifier learning.

4 Methodology

Overview. Consider a pretrained flow-matching VLA parameterized by a velocity field $\mathbf{v}_\pi$. Our objective is to learn a lightweight verifier that is also aware of cross-chunk temporal coherence, while requiring no additional expert demonstrations. To this end, we propose Temporal Verification, whose overall pipeline is illustrated in Figure 2(a). It first learns a temporal token through a dynamics-aware loss that compactly summarizes recent observation-action history. Conditioned on this token, we train an energy-based verifier with a contrastive objective (Figure 2(b)), treating the corresponding expert demonstration as the positive sample and constructing two negative families: temporally mismatched chunks and policy-generated deviations. Finally, we use the learned verifier for gradient-based guidance, steering intermediate candidates toward regions that may yield improved action chunks.

4.1 Temporal Token Learning

Conditioning stateless flow-matching VLAs [2, 16] on dense observation histories can be computationally expensive [37]. We address this by compressing recent history into a compact *temporal token*, providing the verifier with temporal context while maintaining inference efficiency.

Given a history window of length W, we first construct an observation summary for each timestep $i \in \{t-W+1, \ldots, t\}$. Specifically, we pass observation $\mathbf{o}_i$ through the pretrained VLM backbone and extract its token representations. To summarize the observation at timestep i, we project these tokens into a latent space of dimension d_h and append a learnable readout token shared across timesteps. A lightweight transformer processes the projected tokens with bidirectional attention, and the output at the readout position is used as the compressed observation summary, denoted by $\mathbf{h}_i \in \mathbb{R}^{d_h}$. This step produces compact per-timestep summaries, but each $\mathbf{h}_i$ still represents its observation independently and does not yet encode temporal evolution.

To incorporate execution history, we pair each observation summary $\mathbf{h}_i$ with the immediately preceding executed action $\mathbf{a}_{i-1}$. We concatenate $\mathbf{h}_i$ and $\mathbf{a}_{i-1}$, and feed the resulting sequence into a temporal transformer $\mathcal{T}_\psi$ with a causal attention mask [39, 5]:

$$(\boldsymbol{\tau}_{t-W+1}, \ldots, \boldsymbol{\tau}_t) = \mathcal{T}_\psi([\mathbf{h}_{t-W+1}; \mathbf{a}_{t-W}], \ldots, [\mathbf{h}_t; \mathbf{a}_{t-1}]) \,. \tag{3}$$

The final output $\boldsymbol{\tau}_t$ causally aggregates information from the recent observation-action history. We define $\boldsymbol{\tau}_t$ as the **temporal token**, a compact representation of the recent execution trajectory used to condition the verifier for candidate chunks generated at time t.

To encourage the temporal token to capture coherent state-action evolution, we train it with an auxiliary future-action prediction objective. Let $g(\cdot)$ denote a lightweight prediction head that maps $\boldsymbol{\tau}_t$ to F future actions. We define the auxiliary loss as

$$\mathcal{L}_\tau = \left\| g(\boldsymbol{\tau}_t) - [\mathbf{a}_t, \ldots, \mathbf{a}_{t+F-1}] \right\|^2, \tag{4}$$

where $[\mathbf{a}_t, \ldots, \mathbf{a}_{t+F-1}]$ denotes the ground-truth future action sequence from the offline trajectory. This auxiliary objective acts as a dynamics-aware regularizer: it encourages $\boldsymbol{\tau}_t$ to retain information that links recent observations and executed actions to plausible future motion, making it better suited for temporally conditioned action ranking and steering.

4.2 Contrastive Verifier Training

We next introduce a lightweight energy model E_θ that serves as a temporally aware verifier. Conditioned on the temporal token $\boldsymbol{\tau}_t$, it assigns an energy score to each sampled action chunk candidate under the same recent execution context. Because verification only requires relative preference among candidates rather than calibrated reward estimation, we formulate verifier learning as a contrastive ranking problem, encouraging expert chunks to receive lower energy than inferior alternatives.

For each training timestep t, let $\mathbf{z}_t^+$ denote the latent representation of the expert action chunk. We use $\mathbf{z}_t^+$ as the positive sample, reflecting the supervision available from offline imitation data. We then construct two classes of negative samples, each designed to expose a distinct failure mode of generative action prediction. Contrasting positives against these negatives trains the verifier to capture the compatibility criteria required for temporally aware candidate ranking and guidance.

❶ **Temporal-mismatch negatives.** Diffusion- and flow-matching-based policies can model multimodal action distributions [8]. However, this expressiveness can also induce mode switching during execution: consecutive chunks may appear individually plausible while remaining mutually inconsistent, resulting in jittery trajectories, mode oscillation, and task failure [24, 3]. We therefore construct *temporal-mismatch negatives* $\{\mathbf{z}^-_{t,\mathrm{mis},k}\}_{k=1}^{K_{\mathrm{mis}}}$ to train the verifier to recognize temporal compatibility. For a positive chunk $\mathbf{z}_t^+$, we sample expert chunks from other timesteps j within the same episode, subject to a minimum temporal distance $|j - t| \geq \delta$. Although these chunks are often high quality, they are not the correct continuation of the history encoded by $\boldsymbol{\tau}_t$. Contrasting them against $\mathbf{z}_t^+$ therefore encourages the verifier to distinguish temporally compatible chunks from plausible but mismatched alternatives. This construction is particularly useful under latent-state ambiguity, where different physical states induce similar representations. In such cases, action chunks from other timesteps in the same episode may appear semantically plausible while remaining temporally misaligned with the robot's recent execution.

❷ **Coarse-integration negatives.** Temporal-mismatch negatives capture expert-like chunks that are incompatible with the current history, but they do not expose the verifier to errors produced by the policy itself. We therefore construct *coarse-integration negatives* $\{\mathbf{z}^-_{t,\mathrm{crs},k}\}_{k=1}^{K_{\mathrm{crs}}}$ from policy-generated chunks produced under the same observation context but with fewer Euler steps $n < N$. These negatives arise from a coarser numerical approximation of the learned flow and are therefore likely to be lower quality than full-step samples. We draw negatives across different values of n: smaller n yields easier negatives, while larger n produces harder negatives that more closely resemble standard policy outputs. This encourages the verifier to distinguish high-quality generated chunks from suboptimal policy samples.

Having defined the negative families, we train the verifier with a margin-based contrastive objective. Let $\mathcal{R} \coloneqq \{\mathrm{mis}, \mathrm{crs}\}$ denote the set of negative types. For each family $r \in \mathcal{R}$, let $\{\mathbf{z}^-_{t,r,k}\}_{k=1}^{K_r}$ denote the negative samples constructed for training timestep t, and let $m_r > 0$ denote the corresponding margin. The contrastive verifier loss is defined as

$$\mathcal{L}_c = \sum_{r \in \mathcal{R}} \frac{1}{K_r} \sum_{k=1}^{K_r} \Big[m_r + E_\theta\big(\mathbf{z}_t^+ \mid \boldsymbol{\tau}_t\big) - E_\theta\Big(\mathbf{z}^-_{t,r,k} \mid \boldsymbol{\tau}_t\Big) \Big]_+, \tag{5}$$

where $[x]_+ \coloneqq \max\{x, 0\}$. Since lower energy indicates higher compatibility, minimizing $\mathcal{L}_c$ encourages the expert chunk $\mathbf{z}_t^+$ to receive lower energy than each negative sample by at least the corresponding margin m_r. Although these negative constructions are not exhaustive, they provide simple and effective supervision for test-time verification.

Table 1: **Performance (%) comparison on LIBERO-Plus.** Best performance in **bold**.

Method	*Camera*	*Robot*	*Language*	*Light*	*Background*	*Noise*	*Layout*	Average
Base [16]	41.8	72.3	79.9	82.8	84.4	76.2	84.9	73.2
Random	43.9	70.2	86.7	81.8	85.5	79.3	81.1	74.3
TE [50]	44.4	72.3	82.2	76.3	85.1	74.8	84.0	73.0
BID [24]	44.2	73.3	76.2	81.4	80.3	76.6	81.7	72.2
TACO [46]	44.9	**77.1**	90.1	77.4	86.2	77.1	87.2	76.0
TACO* [46]	42.7	74.8	83.8	76.6	76.8	72.2	80.1	71.5
Ours	**48.4**	75.6	**91.9**	**84.7**	**95.8**	**80.0**	**92.6**	**79.8**

Finally, we jointly optimize the temporal encoder and verifier using

$$\mathcal{L} = \mathcal{L}_c \; + \; \lambda_\tau \mathcal{L}_\tau, \tag{6}$$

where λ_τ controls the weight of the auxiliary temporal-token prediction loss. The prediction head used for $\mathcal{L}_\tau$ is discarded after training.

4.3 Verification as Guidance

The learned verifier defines a differentiable energy function over action chunk representations conditioned on recent execution history. This enables the verifier to serve not only as a post-hoc ranking module, but also as a guidance signal during flow integration [9]. Let $\mathbf{z}_s$ denote the differentiable action representation extracted at flow time s. Since $\mathbf{z}_s$ depends on the intermediate action sample $\mathbf{x}_s$, the verifier induces a gradient $\nabla_{\mathbf{x}_s} E_\theta(\mathbf{z}_s \mid \boldsymbol{\tau}_t)$ through the action expert. For each sampled candidate m, we initialize an independent Gaussian noise sample $\mathbf{x}_0^{(m)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. The VLA prefix key-value cache $\mathcal{K}_t$ is computed once from the current observation and shared across candidates. We then augment the standard Euler update in Eq. 1 with an energy-descent term:

$$\mathbf{x}_{s+\frac{1}{N}}^{(m)} = \mathbf{x}_s^{(m)} + \frac{1}{N}\,\mathbf{v}_\pi\left(\mathbf{x}_s^{(m)}, s \mid \mathcal{K}_t\right) - \frac{\eta_s}{N}\nabla_{\mathbf{x}_s^{(m)}} E_\theta\left(\mathbf{z}_s^{(m)} \mid \boldsymbol{\tau}_t\right), \tag{7}$$

where $\eta_s \geq 0$ controls the guidance strength at flow time s. We linearly decay the guidance strength from $\eta_0 = \eta$ to zero over the integration process, making guidance stronger in early steps and weaker near the end. This schedule allows the verifier to provide coarse global correction while the sample is still noisy, while preserving the base policy's fine-grained refinement in later Euler steps.

After integration, each candidate yields a final action chunk $\mathbf{x}_1^{(m)}$ and final representation $\mathbf{z}_1^{(m)}$. We select the executed chunk by minimum verifier energy:

$$m^* = \underset{m \in \{1,\ldots,M\}}{\arg\min}\; E_\theta\left(\mathbf{z}_1^{(m)} \mid \boldsymbol{\tau}_t\right), \tag{8}$$

and execute $\mathbf{x}_1^{(m^*)}$. The verifier therefore plays a dual role: during integration, it steers intermediate samples toward lower-energy regions of the action space; after integration, it ranks completed candidates for execution. Algorithm 1 in the Appendix summarizes the full procedure.

5 Experiments

We evaluate TeV in both simulation and real-world settings across diverse perturbations and manipulation tasks. In simulation, we follow the standard evaluation protocol for each benchmark. In addition to comparisons with baseline methods, we conduct diagnostic analyses to isolate the contribution of each component and better understand the mechanisms behind the observed performance gains.

5.1 Simulation Environment

We evaluate on two simulation benchmarks: LIBERO-Plus [12] and RoboTwin 2.0 [6]. LIBERO-Plus contains 10,030 tasks spanning seven perturbation types: object layout (*Layout*), camera viewpoint (*Camera*), robot initial state (*Robot*), language instruction (*Language*), lighting condition (*Light*), background texture (*Background*), and sensor noise (*Noise*). These variations allow us to assess

Table 2: **Performance (%) comparison on RoboTwin2.0 tasks.** Best performance in **bold**.

Method	*Adjust Bottle*	*Pick Bottles*	*Place Container*	*Stack Bowls*	*Place Cup*	*Open Laptop*	*Press Stapler*	Average
Base [16]	86.0	48.0	86.0	89.0	78.0	75.0	44.0	72.3
Random	85.0	52.0	90.0	88.0	79.0	72.0	50.0	73.7
TE [50]	87.0	43.0	86.0	88.0	82.0	74.0	47.0	72.4
BID [24]	87.0	59.0	87.0	91.0	76.0	77.0	47.0	74.9
TACO [46]	81.0	46.0	87.0	85.0	80.0	62.0	49.0	70.0
TACO* [46]	85.0	32.0	85.0	87.0	80.0	67.0	52.0	69.7
Ours	**95.0**	**64.0**	**94.0**	**93.0**	**88.0**	**81.0**	**54.0**	**81.3**

policy robustness under diverse visual, physical, and linguistic shifts. For RoboTwin 2.0, we select seven representative manipulation tasks: *adjust bottle position*, *pick dual bottles*, *place container onto plate*, *stack two bowls*, *place empty cup on coaster*, *open laptop*, and *press stapler*. For each task, we collect 50 expert demonstrations for policy training. During evaluation, each task is repeated for 100 trials, and we report task success rate as the primary metric.

Baselines. We compare TeV against verification-based and temporal-consistency baselines, all using the same $\pi_{0.5}$ [16] policy backbone and checkpoint. Unless otherwise specified, all test-time scaling methods use the same candidate budget, sampling $M = 4$ action chunks per observation.

- **Random selection**: This baseline randomly selects one candidate chunk from the sampled set for execution, isolating the effect of multi-candidate sampling without any verification signal.
- **Temporal Ensembling (TE)** [50]: This baseline maintains a buffer of previously predicted action chunks and executes a weighted average of the actions assigned to the current timestep, smoothing execution across overlapping chunks and enhances temporal consistency.
- **Bidirectional Decoding (BID)** [24]: BID samples multiple candidate chunks and selects the one most similar with the previously executed chunk over overlapping timesteps.
- **TACO** [46]: TACO uses a lightweight Coin-Flipping Network to estimate pseudo-counts over internal policy representations, and uses these estimates to prefer action chunks closer to high-density regions of the training distribution. We also report **TACO***, which uses $M = 50$ candidates per observation, matching the default candidate budget in the original paper.

Table 3: **Performance (%) over different M.**

Task	M=1	M=2	M=4	M=8	M=16
LIBERO-Plus	55.9	62.0	79.8	79.8	77.5
Adjust Bottle	76.0	84.0	95.0	88.0	87.0
Press Stapler	52.0	48.0	54.0	57.0	58.0

Results. Tables 1 and 2 report the performance comparisons. Temporal Ensembling and BID show mixed effects, often performing similarly to the base policy and random selection. This suggests that directly applying existing temporal-consistency baselines to these simulation benchmarks is insufficient. TACO improves performance, whereas TACO* underperforms by a large margin, indicating that increasing the candidate budget does not necessarily lead to higher success rates. By contrast, TeV delivers more consistent improvements across the evaluated task suites, achieving average gains of 3.8% over the strongest baseline on LIBERO-Plus and 6.4% on RoboTwin 2.0 subtasks. These results indicate that verifier design is critical for translating additional test-time samples into performance gains.

The limited gains of temporal-consistency baselines may stem from the relatively static task setups and simplified physical constraints in simulation. Both TE and BID refine the current prediction using only consecutive action chunks. In these benchmarks, however, the predicted chunks are often already clean and high quality due to well-collected expert demonstrations. This leaves limited room for smoothing-based improvement and similarity-based chunk selection. TACO performs less effectively on RoboTwin 2.0, possibly due to the limited number of expert demonstrations used to train the base policy. With sparse training coverage, pseudo-count-based estimates of training-distribution support may become less reliable. In contrast, TeV incorporates recent observation-action history and trains the verifier contrastively, providing a richer context for candidate evaluation and broader supervision over plausible but mismatched action chunks.

Ablations. We conduct the following ablation studies to isolate the contribution of individual components and assess the effect of key hyperparameters.

- **Guidance and verification.** Figure 3 compares three variants: verification-only, guidance-only, and the full method, while also varying the guidance strength η. Verification-only consistently

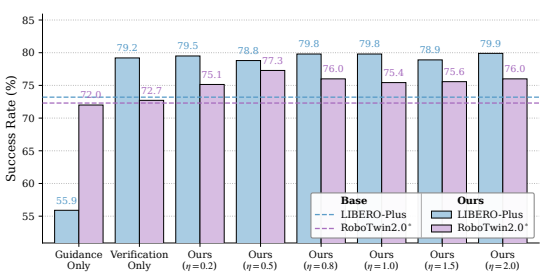

Figure 3: **Effect of guidance and verification.**

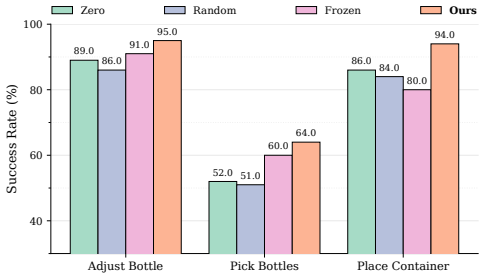

Figure 4: **Effect of temporal token variants.**

improves task success, showing that the learned energy model provides a reliable signal for candidate ranking. Guidance-only degrades performance on both benchmarks, suggesting that gradient-based steering is unstable without final candidate selection. If guidance pushes a candidate toward an unfavorable region of the action space, the resulting deviation cannot be rejected and may compound during execution. Combining guidance with final candidate ranking yields the strongest overall performance. Across the tested values of η, the full method generally outperforms the baselines and remains competitive in the remaining cases, suggesting that Temporal Verification is relatively robust to the guidance strength value.

- **Effect of negative construction.** We evaluate the role of each negative family by training verifiers with different negative combinations on LIBERO-Plus. The base policy achieves a success rate of 72.3%. Training with only coarse-integration negatives improves success to 74.3%, while training with only temporal-mismatch negatives yields a larger gain, reaching 79.2%. These results indicate that both negative types are useful, with temporal-mismatch negatives providing the larger contribution. Combining the two families further improves success to 79.8%, suggesting that coarse-integration negatives add complementary supervision beyond temporal compatibility.
- **Effect of the number of candidates.** Table 3 examines how the candidate budget affects performance. Task success does not increase monotonically with the number of sampled candidates, indicating that additional samples are not automatically beneficial. This is expected, since an imperfect verifier may face harder ranking decisions as larger candidate sets include more out-of-distribution or low-quality chunks.
- **Effect of the temporal token.** Figure 4 evaluates the learned temporal token against three variants: ***Zero***, which replaces it with **0**; ***Random***, which replaces it with a Gaussian random vector; and ***Frozen***, which reuses the first temporal token computed in each episode for all subsequent steps. The learned temporal token consistently outperforms all three variants. Improvements over the *Zero* and *Random* baselines confirm the effectiveness of temporal-token conditioning, while the gain over *Frozen* shows that dynamically updating the token with recent observation-action history provides meaningful context for verifier-based ranking and steering.

5.2 Real-world Environment

Setup. We consider three tasks: (1) put water bottle into the bag (*Bottle to Bag*), (2) move the water cup from the shelf to the table (*Cup Transport*), and (3) pour water from the right cup into the left cup (*Water Filling*). We collect 180 trajectories in total for policy training. During execution, the robot runs at 15Hz with a fixed execution horizon. We compare TeV against the base policy, Bidirectional Decoding [24], and TACO [46]. For execution safety, all methods adopt Temporal Ensembling [50]. Test-time scaling methods sample $M = 4$ candidates per observation. We evaluate performance using a task completion score, assigning each episode an integer score according to the successfully completed substeps. Each setting is evaluated over 30 trials with varied object positions.

Results. Table 4 reports the average task completion scores, including mean and standard deviation, across methods and tasks. Different from simulation, these real-world tasks require stable execution and smooth movement: the bottle in *Bottle to Bag* is deformable, while the other two tasks involve transporting or pouring liquid. Across the baselines, we observe occasional trajectory oscillations and unstable execution, resulting in failures such as dropped bottles or spilled water. In contrast, TeV produces more stable and consistent trajectories, reducing failures associated with jerky motion and achieving stronger overall performance. Baseline methods also sometimes terminate water pouring prematurely, before fully emptying the source cup. This may be due to partial observability during cup tilting, as neither the wrist camera nor the third-view camera reliably captures the liquid state

Table 4: **Performance (%) on real-world tasks.** Best results marked in **bold**.

Method	*Bottle to Bag*	*Cup Transport*	*Water Filling*
Base [16]	60.7 ± 31.9	79.8 ± 27.5	40.5 ± 32.4
BID [24]	59.8 ± 27.8	71.5 ± 25.9	51.3 ± 30.5
TACO [46]	64.0 ± 30.5	75.2 ± 21.2	51.5 ± 29.7
Ours	**69.3** ± 26.7	**88.3** ± 15.9	**68.0** ± 12.8

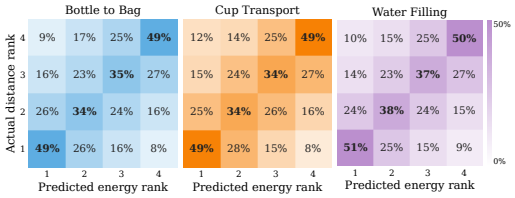

Figure 5: **Energy score reliability.**

inside the cups throughout the motion. By conditioning verification on encoded temporal history, TeV can use recent execution context as an additional signal of task progress, which may help maintain a consistent trajectory when the liquid state is only partially observable.

Analysis. Figure 5 evaluates the reliability of the predicted energy ranking. We compare the verifier-induced ranking of sampled candidates with their latent-space distance ranking to the corresponding expert action chunk. The verifier selects one of the two nearest candidates in more than 70% of cases, indicating that lower-energy candidates generally align more closely with expert behavior. This supports the effectiveness of the contrastive verifier learning objective.

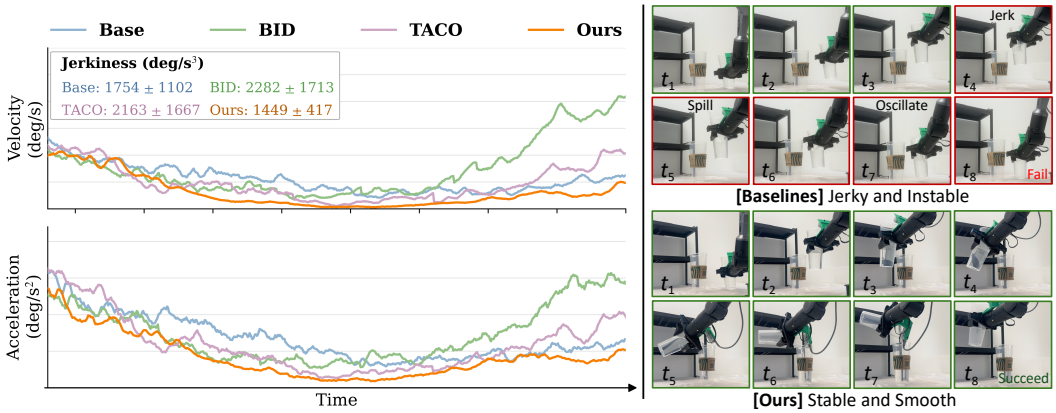

Figure 6: **Illustration of robot kinematics and task progress over time. Left:** Average joint velocity and acceleration during interaction with the water cup. **Right:** Real-world rollouts. The baselines often exhibit abrupt motions and oscillations, leading to water spilling or task failure, whereas our method maintains a steadier trajectory and completes the task without spilling.

Figure 6 analyzes average robot kinematics and task progress rate. On the left, we illustrate metrics including joint velocity and acceleration, and quantify jerkiness as the per-trial mean magnitude of the third time derivative of joint positions, where lower values indicate fewer abrupt motion changes. The results show that TeV produces smoother kinematic profiles with lower jerkiness, which is associated with fewer failures such as water spilling. This supports its effectiveness in improving execution smoothness and temporal consistency. On the right, we provide example rollouts showing baseline failure cases, such as oscillatory motion and water spilling, which are not observed in the corresponding TeV rollouts. Additional rollout videos are provided in the supplementary.

6 Conclusion

Verification-based test-time scaling methods often suffer from temporal myopia, overlooking the consistency required across consecutive action chunks. In this paper, we introduced Temporal Verification (TeV), an efficient energy-based framework that makes action verification temporally aware. TeV equips a lightweight energy-based verifier with a compact temporal token, enabling candidate actions to be evaluated against recent observation-action history rather than a static timestep. Since verification requires relative preference among candidates rather than calibrated reward estimation, we train the verifier with a contrastive objective. The resulting verifier serves two roles: it ranks completed candidates for selection and actively steers intermediate policy outputs during generation. Extensive experiments in simulation and real-world manipulation tasks show that TeV is sample-efficient, improves success rates, and yields smoother robot dynamics.

References

[1] Yusuf Ali, Gryphon Patlin, Karthik Kothuri, Muhammad Zubair Irshad, Wuwei Liang, and Zsolt Kira. Eve: A generator-verifier system for generative policies, 2025.

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

[3] Kevin Black, Manuel Y. Galliker, and Sergey Levine. Real-time execution of action chunking flow policies, 2025.

[4] Haoyuan Cai, Zhenghao Peng, and Bolei Zhou. Predictive preference learning from human interventions, 2025.

[5] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling, 2021.

[6] Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, Weiliang Deng, Yubin Guo, Tian Nian, Xuanbing Xie, Qiangyu Chen, Kailun Su, Tianling Xu, Guodong Liu, Mengkang Hu, Huan ang Gao, Kaixuan Wang, Zhixuan Liang, Yusen Qin, Xiaokang Yang, Ping Luo, and Yao Mu. Robotwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation, 2025.

[7] Xinyi Chen, Yilun Chen, Yanwei Fu, Ning Gao, Jiaya Jia, Weiyang Jin, Hao Li, Yao Mu, Jiangmiao Pang, Yu Qiao, Yang Tian, Bin Wang, Bolun Wang, Fangjing Wang, Hanqing Wang, Tai Wang, Ziqin Wang, Xueyuan Wei, Chao Wu, Shuai Yang, Jinhui Ye, Junqiu Yu, Jia Zeng, Jingjing Zhang, Jinyu Zhang, Shi Zhang, Feng Zheng, Bowen Zhou, and Yangkun Zhu. Internvla-m1: A spatially guided vision-language-action framework for generalist robot policy, 2025.

[8] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion, 2024.

[9] Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis, 2021.

[10] Maximilian Du and Shuran Song. Dynaguide: Steering diffusion polices with active dynamic guidance, 2025.

[11] Yilun Du and Igor Mordatch. Implicit generation and modeling with energy based models. *Advances in neural information processing systems*, 32, 2019.

[12] Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, Jinlan Fu, Jingjing Gong, and Xipeng Qiu. Libero-plus: In-depth robustness analysis of vision-language-action models, 2025.

[13] Raia Hadsell, Sumit Chopra, and Yann Lecun. Dimensionality reduction by learning an invariant mapping. pages 1735 – 1742, 02 2006.

[14] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE, 2006.

[15] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels, 2019.

[16] Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization, 2025.

[17] Wenqi Jiang, Jason Clemons, Karu Sankaralingam, and Christos Kozyrakis. How fast can i run my vla? demystifying vla inference performance with vla-perf, 2026.

[18] Taeyoung Kim, Jimin Lee, Myungkyu Koo, Dongyoung Kim, Kyungmin Lee, Changyeon Kim, Younggyo Seo, and Jinwoo Shin. Contrastive representation regularization for vision-language-action models, 2025.

[19] 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, 2025.

[20] 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, 2026.

[21] Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, Fujie Huang, et al. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

[22] Yishu Li, Xinyi Mao, Ying Yuan, Kyutae Sim, Ben Eisner, and David Held. Learn from what we have: History-aware verifier that reasons about past interactions online, 2025.

[23] Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

[24] Yuejiang Liu, Jubayer Ibn Hamid, Annie Xie, Yoonho Lee, Max Du, and Chelsea Finn. Bidirectional decoding: Improving action chunking via guided test-time sampling. *International Conference on Learning Representations (ICLR)*, 2025.

[25] Mingyang Lyu, Yinqian Sun, Erliang Lin, Huangrui Li, Ruolin Chen, Feifei Zhao, and Yi Zeng. Reinforcement fine-tuning of flow-matching policies for vision-language-action models, 2025.

[26] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks, 2018.

[27] Mitsuhiko Nakamoto, Oier Mees, Aviral Kumar, and Sergey Levine. Steering your generalists: Improving robotic foundation models via value guidance, 2025.

[28] Kensuke Nakamura, Lasse Peters, and Andrea Bajcsy. Generalizing safety beyond collision-avoidance via latent-space reachability analysis. In *Robotics: Science and Systems XXI*. Robotics: Science and Systems Foundation, June 2025.

[29] NVIDIA, :, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi "Jim" Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, Kevin Lin, Guilin Liu, Edith Llontop, Loic Magne, Ajay Mandlekar, Avnish Narayan, Soroush Nasiriany, Scott Reed, You Liang Tan, Guanzhi Wang, Zu Wang, Jing Wang, Qi Wang, Jiannan Xiang, Yuqi Xie, Yinzhen Xu, Zhenjia Xu, Seonghyeon Ye, Zhiding Yu, Ao Zhang, Hao Zhang, Yizhou Zhao, Ruijie Zheng, and Yuke Zhu. Gr00t n1: An open foundation model for generalist humanoid robots, 2025.

[30] Zhenghao Peng, Wenjie Mo, Chenda Duan, Quanyi Li, and Bolei Zhou. Learning from active human involvement through proxy value propagation, 2025.

[31] Stephane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning, 2011.

[32] Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, Simon Alibert, Matthieu Cord, Thomas Wolf, and Remi Cadene. Smolvla: A vision-language-action model for affordable and efficient robotics, 2025.

[33] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters, 2024.

[34] Yunzhou Song, Long Le, Yong-Hyun Park, Jie Wang, Junyao Shi, Lingjie Liu, Jiatao Gu, Eric Eaton, Dinesh Jayaraman, and Kostas Daniilidis. Omniguide: Universal guidance fields for enhancing generalist robot policies, 2026.

[35] Zhanyi Sun and Shuran Song. Latent policy barrier: Learning robust visuomotor policies by staying in-distribution, 2025.

[36] Jiaming Tang, Yufei Sun, Yilong Zhao, Shang Yang, Yujun Lin, Zhuoyang Zhang, James Hou, Yao Lu, Zhijian Liu, and Song Han. Vlash: Real-time vlas via future-state-aware asynchronous inference, 2025.

[37] Marcel Torne, Karl Pertsch, Homer Walke, Kyle Vedder, Suraj Nair, Brian Ichter, Allen Z. Ren, Haohuan Wang, Jiaming Tang, Kyle Stachowicz, Karan Dhabalia, Michael Equi, Quan Vuong, Jost Tobias Springenberg, Sergey Levine, Chelsea Finn, and Danny Driess. Mem: Multi-scale embodied memory for vision language action models, 2026.

[38] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding, 2019.

[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[40] Haoxuan Wang, Gengyu Zhang, Yan Yan, Yuzhang Shang, Ramana Rao Kompella, and Gaowen Liu. Real-time robot execution with masked action chunking, 2026.

[41] Philipp Wu, Alejandro Escontrela, Danijar Hafner, Ken Goldberg, and Pieter Abbeel. Daydreamer: World models for physical robot learning, 2022.

[42] Wei Wu, Fan Lu, Yunnan Wang, Shuai Yang, Shi Liu, Fangjing Wang, Qian Zhu, He Sun, Yong Wang, Shuailei Ma, Yiyu Ren, Kejia Zhang, Hui Yu, Jingmei Zhao, Shuai Zhou, Zhenqi Qiu, Houlong Xiong, Ziyu Wang, Zechen Wang, Ran Cheng, Yong-Lu Li, Yongtao Huang, Xing Zhu, Yujun Shen, and Kecheng Zheng. A pragmatic vla foundation model, 2026.

[43] Yilin Wu, Ran Tian, Gokul Swamy, and Andrea Bajcsy. From foresight to forethought: Vlm-in-the-loop policy steering via latent alignment, 2025.

[44] Wenli Xiao, Haotian Lin, Andy Peng, Haoru Xue, Tairan He, Yuqi Xie, Fengyuan Hu, Jimmy Wu, Zhengyi Luo, Linxi "Jim" Fan, Guanya Shi, and Yuke Zhu. Self-improving vision-language-action models with data generation via residual rl, 2025.

[45] Charles Xu, Jost Tobias Springenberg, Michael Equi, Ali Amin, Adnan Esmail, Sergey Levine, and Liyiming Ke. RL token: Bootstrapping online RL with vision-language-action models. Technical report, Physical Intelligence, 2025.

[46] 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, 2025.

[47] Checheng Yu, Chonghao Sima, Gangcheng Jiang, Hai Zhang, Haoguang Mai, Hongyang Li, Huijie Wang, Jin Chen, Kaiyang Wu, Li Chen, Lirui Zhao, Modi Shi, Ping Luo, Qingwen Bu, Shijia Peng, Tianyu Li, and Yibo Yuan. χ_0: Resource-aware robust manipulation via taming distributional inconsistencies, 2026.

[48] Hongyin Zhang, Shuo Zhang, Junxi Jin, Qixin Zeng, Runze Li, and Donglin Wang. Robustvla: Robustness-aware reinforcement post-training for vision-language-action models, 2025.

[49] Wanpeng Zhang, Ye Wang, Hao Luo, Haoqi Yuan, Yicheng Feng, Sipeng Zheng, Qin Jin, and Zongqing Lu. Dig-flow: Discrepancy-guided flow matching for robust vla models, 2025.

[50] Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware, 2023.

A Implementation Details

Training pipeline. We implement the temporal encoder as a lightweight transformer-style module with 3.0M parameters. The energy verifier is a shallow feedforward network with 0.9M parameters, equipped with spectral normalization [26] that constrains the verifier's Lipschitz constant and stabilizes energy gradients. Together, the temporal encoder and verifier add less than 0.15% parameters relative to the base model. For training-data construction, we uniformly sample from both negative families. The base model remains frozen throughout training, while only the added modules are optimized from scratch.

Deployment framework. During deployment, we maintain a rolling buffer of recent observations and executed actions to compute the temporal token $\boldsymbol{\tau}_t$. At the first decision step, when no action history is available, we fall back to standard flow-matching generation; at later steps, any remaining missing action entries are zero-padded. To reduce test-time scaling latency, we use two optimizations. First, we compute the VLA key-value cache $\mathcal{K}_t$ once from the current observation and share it across all M candidates, avoiding repeated backbone computation. Second, we use an efficient inference variant with *guidance cutoff*. Verifier guidance is applied only during the first $\lfloor N/2 \rfloor$ Euler steps, after which candidates are scored and the $\lceil M/2 \rceil$ lowest-energy candidates are retained. The remaining integration steps are completed without guidance. This reduces gradient computation by half and decreases later-stage candidate updates with negligible accuracy degradation.

B Experimental Details

Inference acceleration. Test-time scaling with multiple sampled candidates increases inference latency, which can make real-world execution jerky and instable, leading to abrupt movements that degrade task performance. To reduce this overhead, we use two acceleration strategies. First, we reuse the VLA key-value cache across candidates, since all candidates are conditioned on the same visual-language context, and compile the action-sampling function to reduce execution overhead. Second, we apply progressive pruning: after the early guided integration steps, we retain only the lowest-energy candidates and complete the remaining integration with a smaller candidate set.

Following [32], we measure latency as the elapsed time between observation acquisition and the availability of the corresponding action chunk in the control queue. On an RTX 5080 inference device with $M = 4$ candidates, KV-cache reuse reduces latency by approximately 20–30 ms, compilation reduces latency by approximately 10 ms, and progressive pruning reduces latency by approximately 30–40 ms. For fair comparison, we apply the same applicable acceleration techniques to all baselines and to TeV; progressive pruning is only used when the method supports candidate scoring during generation.

Computational cost. We train the verifier on RTX A6000 Ada GPUs. The training cost depends on the dataset size because negative construction is performed from the available offline trajectories. For the real-world tasks, verifier training takes approximately 0.6 GPU hours, which is significantly smaller compared to methods that rely on large-scaled datasets and models.

Hardware setup. Figure 7 illustrates the real-world hardware setup. We use a master–slave arm framework for demonstration collection and deploy the learned policy on the slave arm. The system uses two cameras for observation and action prediction: a third-view camera that captures the overall workspace and a wrist-mounted first-view camera that provides close-range visual cues for precise manipulation. During execution, the mobile base is fixed to ensure stable and repeatable operation.

Task execution settings. We evaluate on an AgileX Piper 6-DoF arm equipped with a wrist-mounted Intel RealSense D435i camera and a third-view ZED camera. During real-world execution, we use a fixed execution horizon of 8 for all tasks. The smoothing coefficient for temporal ensembling is set to 0.05. We adopt a synchronous execution framework to ensure compatibility across baseline methods, since asynchronous execution may introduce additional latency- and scheduling-related factors that confound performance evaluation.

For each task, we evaluate progress using a 100-point completion score defined as follows:

- *Bottle to Bag*: grasp the water bottle (20), transport the bottle to the target position (50), and successfully drop the bottle into the bag (30).

Figure 7: Robot platform setup.

- *Water Transport*: grasp the cup of water (30), move the cup stably away from the shelf (35), and place the cup on the table (35). We subtract 10 points each time water is spilled or the cup collides with the shelf.
- *Pour Water*: grasp the right cup of water (10), move it stably to the target position (40), pour water into the left cup (40), and place the emptied cup back on the table (10). We subtract 10 points if water is spilled during execution.

Kinematics calculation. For the kinematic analysis on the *Water Filling* task, we compute the metrics using joints J1–J5. We exclude J6 because its motion is dominated by the cup-tilting action required for pouring and is therefore less informative about the stability of the arm trajectory. In contrast, joints J1–J5 primarily determine how steadily the cup is positioned and transported during manipulation. For visualization, we smooth the per-timestep velocity and acceleration magnitudes; all quantitative jerkiness metrics are computed from the original unsmoothed joint trajectories.

C Algorithm

We show the inference process of our method in Algorithm 1.

D Additional Results

Contrastive Loss Comparison Table 5 compares different contrastive objectives for verifier training. In addition to the margin-based contrastive loss used in the main experiments, we also evaluate InfoNCE [38]. Although InfoNCE performs worse than the margin-based objective, it is used without hyperparameter tuning and still yields substantial gains over the base policy and all baselines. This suggests that the performance improvement is not tied to a specific contrastive loss, while the margin-based formulation currently provides the strongest results in our setting.

Energy reliability analysis. Figure 8 reports the energy-ranking reliability analysis on the LIBERO-Plus simulation benchmark. The training dataset is built from large-scale, high-quality demonstrations, and the base policy already produces strong action candidates in many settings. Even in this relatively saturated regime, our verifier provides a meaningful ranking signal: predicted energy is positively

Algorithm 1 Temporal Verification

Require: Velocity field $\mathbf{v}_\pi$, verifier E_θ, temporal encoder $\mathcal{T}_\psi$
Require: Observation history $\{\mathbf{o}_i\}_{i=t-W+1}^{t}$, action history $\{\mathbf{a}_i\}_{i=t-W}^{t-1}$
Require: History length W, candidates M, Euler steps N, guidance schedule $\{\gamma_s\}$
Ensure: Selected action chunk $\mathbf{x}_1^{(m^*)}$

// Encode temporal context
Compute temporal token $\boldsymbol{\tau}_t$ via Eq. (3)
Compute VLA context cache $\mathcal{K}_t$ from $\mathbf{o}_t$
// Verifier-guided candidate generation
for $m = 1, \ldots, M$ **do**
  Sample $\mathbf{x}_0^{(m)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  for $j = 0, \ldots, N-1$ **do**
    $s \leftarrow j/N$
    Extract intermediate representation $\mathbf{z}_s^{(m)}$
    Update $\mathbf{x}_{s+1/N}^{(m)}$ via Eq. (7)
  end for
  Compute final representation $\mathbf{z}^{(m)}$
end for
// Select best candidate
$m^* \leftarrow \arg\min_m E_\theta(\mathbf{z}^{(m)} \mid \boldsymbol{\tau}_t)$ ▷ Eq. (8)
return $\mathbf{x}_1^{(m^*)}$

Table 5: Performance (%) comparison on LIBERO-Plus under different contrastive losses.

Method	*Camera*	*Robot*	*Language*	*Light*	*Background*	*Noise*	*Layout*	Average
InfoNCE	45.3	**76.8**	88.5	83.6	87.5	77.7	87.8	76.9
Margin-based	**48.4**	75.6	**91.9**	**84.7**	**95.8**	**80.0**	**92.6**	**79.8**

correlated with latent-space distance to the corresponding expert action chunk, with lower-energy candidates generally closer to expert behavior.

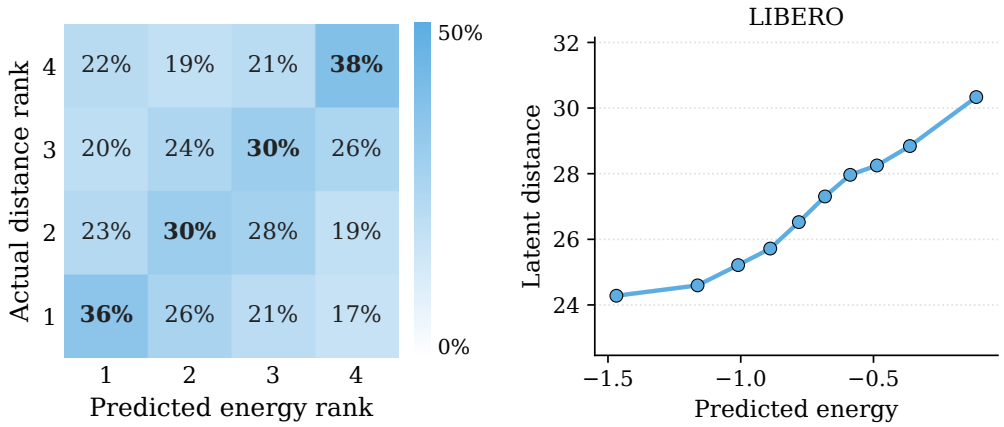

Figure 8: Energy reliability analysis on LIBERO-Plus. **Left:** Rank correlation. **Right:** Energy v.s. Latent Distance.

Experiment rollouts. In Figure 9, we include rollouts from TeV to demonstrate the experimental settings and provide a better intuition of how the tasks are done. For the *Water Filling*, we included a case where the wrist camera was blurred.

Evidence of Temporal Consistency Enhancement Table 6 reports temporal consistency across methods on the *Adjust Bottle* task. We evaluate consistency for arm pose changes, excluding the

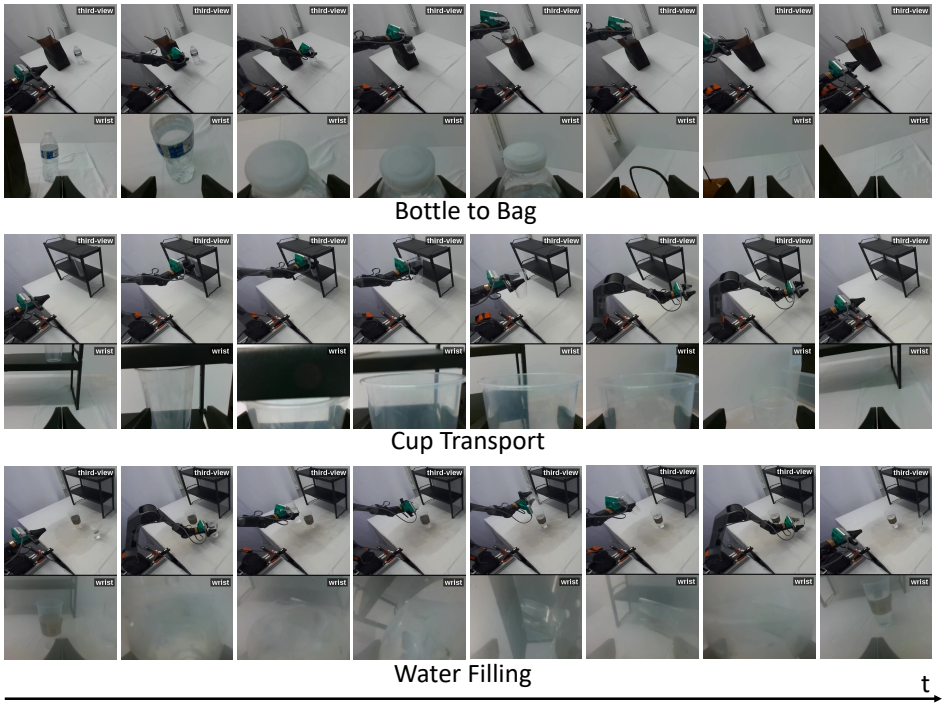

Figure 9: Three real-world tasks rollouts.

gripper, and for the full robot action. Following [24], we measure temporal consistency using the overlap distance between consecutive action chunks. At each decision step, the policy outputs an action chunk of length H, and the next chunk overlaps with the previous one over $H - 1$ timesteps. The overlap distance is computed as the ℓ_2 distance between the two chunks over this shared region in joint space. For each rollout, we compute the 95th percentile of overlap distances across all overlap windows and report the average across rollouts. This metric captures large plan revisions that are likely to affect the downstream controller. Lower values indicate stronger temporal consistency.

Results show that TeV achieves the second-best temporal consistency among the compared methods. BID performs best because it explicitly optimizes for overlap consistency, selecting the candidate with the smallest discrepancy from the previous chunk over shared timesteps. In contrast, TeV does not directly optimize this metric, yet still substantially improves temporal consistency over the base policy and other baselines. This suggests that temporally-aware verification encourages smoother cross-chunk behavior even without explicitly minimizing the overlap-distance objective.

Table 6: Temporal consistency comparison.

Method	Base	Random	TACO	BID	**Ours**
Pose	0.098	0.088	0.078	0.073	**0.076**
Full	0.103	0.089	0.080	0.076	**0.078**

E Societal Impacts

This work seeks to improve the reliability of flow-matching policies by adding lightweight, temporally-aware verification at test time. By encouraging temporal consistency across action chunks, the proposed method can reduce abrupt motions, unstable execution, and task failures, which may benefit various robotic manipulation tasks. However, more reliable robotic manipulation also carries potential risks. Improved deployability may accelerate automation in physical labor domains, raising concerns about workforce displacement and unequal access to robotic technologies. The method also does not provide formal safety guarantees. Real-world deployment should therefore include task-specific safety constraints, human oversight, and extensive validation across diverse conditions. We view

this work as a step toward more trustworthy generative robot control, rather than a replacement for comprehensive safety evaluation and responsible deployment practices.

F Limitations

Our method is not without limitations. First, verifier-guided generation introduces additional inference latency because each guided integration step requires computing energy gradients with respect to intermediate action samples. This overhead can slow down closed-loop execution when multiple candidates are sampled. Future work could reduce the cost using efficient gradient approximations, such as simultaneous perturbation stochastic approximation (SPSA), lower-frequency guidance, cached gradients, or lightweight learned guidance surrogates.Second, TeV can sometimes produce more conservative motions than baseline methods. We hypothesize that the history-aware verifier favors action chunks that stay close to temporally consistent trajectories, reducing abrupt or out-of-distribution behavior but also lowering motion magnitude in some cases. This trade-off is beneficial for stability-sensitive tasks such as liquid transfer, but may limit execution speed when faster actions are safe. Future work should explore adaptive guidance strength and task-dependent objectives that balance temporal consistency, stability, and speed.

G LLM Usage

We used LLM to assist with language polishing and presentation refinement. Specifically, it was employed to improve clarity, grammar, and flow of text, as well as to adjust tone for academic writing. All LLM outputs were carefully reviewed and edited by the authors.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes], [No], or [N/A].
- [N/A] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for [N/A]).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will also be asked to include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While [Yes] is generally preferable to [No], it is perfectly acceptable to answer [No] provided a proper justification is given (e.g., error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering [No] or [N/A] is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims justify as they are.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Discussed in the experiment part as well as supplementary material.

Guidelines:

- The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: Not relevant.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Covered in methodology, algorithm, and experiment parts.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Will be made publicly available.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).

- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Detailed in experiment part.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Reported in tables when relevant.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Included in supplementary material.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.

- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: Confirmed to conform.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: Included in the supplementary materials.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: No such risks posed.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.

- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Properly credited.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The data collected follows the standard LeRobot format. Data collectors are all included as paper authors.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: Not relevant.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: Not relevant.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLM used for writing and polishing.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.