# LOOKING BACK TO MOVE FORWARD: TEMPORAL VERIFICATION FOR GENERATIVE ROBOT POLICIES

**Anonymous authors**
Paper under double-blind review

## 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, these policies can generate suboptimal action chunks that compound errors over time, eventually driving the robot into out-of-distribution states from which recovery is difficult. Action verification offers a test-time scaling strategy for mitigating this failure mode by sampling multiple candidate actions and using a verifier to select one for execution. Existing approaches, however, remain temporally myopic and costly to train, evaluating candidates from the current observation alone without accounting for trajectory continuity and often relying on large 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, enabling candidate chunks to be evaluated as continuations of the execution trajectory rather than as isolated predictions. Conditioned on this token, TeV constructs positive–negative pairs without additional expert demonstrations or preference annotations and trains an energy-based verifier contrastively to assign lower energy to higher-quality, trajectory-consistent action chunks. Beyond post-hoc ranking, TeV further uses the learned energy landscape to guide intermediate flow samples toward lower-energy regions, improving candidates before final selection. Extensive experiments in simulation and real-world settings demonstrate that TeV provides reliably ranks action candidates, improves task success rates, and produces smoother execution trajectories.

## 1 INTRODUCTION

Generative policies based on diffusion and flow models have rapidly become a major paradigm in robot learning (Chi et al., 2024; Lipman et al., 2022; Black et al., 2026). By scaling imitation learning over large corpora of human demonstrations, these models have enabled robots to perform increasingly complex skills, including household cleaning (Intelligence et al., 2025), cloth folding (Yu et al., 2026), and contact-rich assembly such as screw driving (Xu et al., 2025). Within this broader trend, Vision-Language-Action (VLA) policies trained with flow-matching objectives (Lipman et al., 2022) have emerged as a promising approach to generalist robot control (Intelligence et al., 2025; Shukor et al., 2025; NVIDIA et al., 2025; Wu et al., 2026; Chen et al., 2025b). These models combine the semantic representations and reasoning capabilities of vision-language backbones with continuous action generation, mapping language instructions, visual observations, and robot states to temporally extended action chunks. Flow-matching further provides a tractable objective for continuous action generation with efficient sampling, making it well suited to scalable robot learning.

Despite this promise, the flexibility of flow-matching policies also introduces a critical vulnerability. When trained on heterogeneous demonstrations, these policies can generate suboptimal action chunks. Small errors can then compound over time, driving the robot away from the demonstrated state–action manifold and into out-of-distribution states from which recovery becomes increasingly difficult. Existing approaches mitigate this issue by retraining or fine-tuning the base policy through reinforcement learning (Lyu et al., 2025; Xiao et al., 2025; Zhang et al., 2025a) or representation regularization (Zhang et al., 2025b; Kim et al., 2025). While effective, these approaches incur substantial training cost and risk degrading the generalist capabilities encoded by the base policy.



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.

Verification-based test-time scaling (Ali et al., 2025; Yang et al., 2025) therefore offers an alternative, improving action selection at deployment time without modifying the base policy. Rather than committing to a single action chunk, these methods use additional inference-time compute to sample multiple candidates from the base policy. A separate verifier then scores the candidates according to learned expert or preference signals and selects the highest-ranked action for execution.

Yet existing verification-based methods face two key limitations. First, they suffer from temporal myopia, generating and evaluating candidate actions based solely on the current observation. For generative policies, successful execution requires not only selecting high-quality candidates for the current observation, but also ensuring that consecutive action chunks form a coherent trajectory. Without cross-chunk consistency, the policy may oscillate between individually plausible modes, resulting in jerky motion, unstable execution, and task failure (Liu et al., 2025; Wang et al., 2026). Effective verification must therefore account for both *local action plausibility* and *temporal consistency* with recent execution history. Second, existing verifier designs impose substantial training and deployment overhead. Existing approaches often rely on large VLM-based verifiers (Kwok et al., 2025; Ali et al., 2025), requiring careful task adaptation and substantial additional supervision. Obtaining such supervision can be expensive, as reward labels, trajectory preferences, or failure traces require additional annotation, rollout collection, or real-world interaction (Ross et al., 2011; Li et al., 2025). Large verifiers further complicate real-world deployment by adding inference overhead to latency-sensitive control loops, particularly under limited onboard compute (Jiang et al., 2026).

To address these limitations, we propose **Te**mporal **V**erification (**TeV**), an efficient energy-based verification framework for flow-matching VLAs. TeV improves both action quality and temporal consistency by combining post-hoc candidate ranking with energy-guided generation. Specifically, we first construct a *temporal token* that summarizes the robot's recent observation–action history. Encoded efficiently and regularized with a dynamics-aware objective, the token provides the verifier with execution context beyond the current observation. Conditioned on this token, we instantiate the verifier as a lightweight energy model that scores sampled action chunks and ranks them for execution. Because candidate selection requires only relative comparison rather than calibrated reward prediction, we train the verifier contrastively using expert demonstration chunks as positives and temporally mismatched or policy-generated chunks as informative negatives. Finally, TeV extends verification beyond post-hoc ranking by using gradients of the learned energy function to guide flow integration, guiding intermediate samples toward lower-energy regions before final selection.

TeV is a lightweight add-on for flow-matching VLAs that evaluates sampled action chunks against recent execution history rather than the current observation alone. It increases the parameter count by less than 0.15%, requires no additional expert demonstrations, and is the first action-verification framework to integrate candidate ranking with generation-time guidance. Extensive simulation and real-world experiments show that TeV reliably ranks action candidates, improves task success by 6–18 percentage points, and produces smoother execution trajectories, demonstrating the benefits of temporally aware verification for generative robot control.

## 2 Related Work

**Test-time Scaling via Action Verification.** Test-time verification improves model performance by allocating additional inference-time computation to evaluate and rank candidate outputs (Snell et al., 2024). In robotics, this paradigm has been explored through different verifier designs. External-verifier methods use vision-language models to evaluate candidate actions or plans (Kwok et al., 2025; Ali et al., 2025). RL-based approaches support verification through online policy adaptation (Li et al., 2025) or learned value functions for action ranking (Nakamoto et al., 2025). Other methods use verification to improve instruction following (Kwok et al., 2026). The most relevant verification approach to ours is TACO (Yang et al., 2025), 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, whereas our method conditions verification on recent execution history to encourage temporally coherent and smooth behavior.

**Temporal Consistency for Chunked Policies.** Flow-matching policies can generate locally plausible yet temporally inconsistent action chunks, causing mode oscillation and jerky behaviors. Existing methods address this by modifying the chunk execution process. ACT (Zhao et al., 2023) smooths execution by averaging overlapping action chunks, while BID (Liu et al., 2025) samples multiple

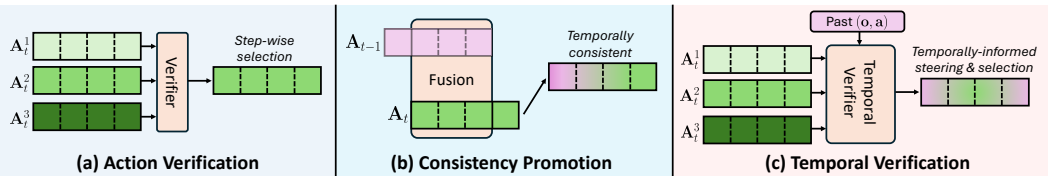


Figure 1: **Paradigm comparison.** **(a)** Conventional action verification selects among multiple sampled candidates at each decision step, relying only on the current observation. **(b)** Temporal-consistency methods refine the current prediction by using previous action chunks to promote cross-chunk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

candidates and selects the chunk most consistent with the previous prediction over overlapping timesteps. Real-time control methods (Black et al., 2025; Tang et al., 2025; Wang et al., 2026) study temporal consistency under asynchronous execution, where stale observations further complicate the problem. In contrast, our method learns a history-conditioned energy verifier that evaluates whether each candidate is an expert-compatible continuation of the recent observation–action trajectory.

**Policy Steering.** Inference-time steering improves pretrained generative policies by modifying the sampling or refinement process while keeping the base policy fixed (Nakamoto et al., 2025). 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 (Nakamura et al., 2025; Song et al., 2026) guide policy outputs using learned dynamics models to satisfy safety constraints or improve task performance, while human-in-the-loop approaches (Cai et al., 2025; Peng et al., 2025) refine actions based on user-provided subgoals, corrections, or preferences. Classifier- and dynamics-guided methods (Du & Song, 2025; Sun & Song, 2025) use latent visual dynamics models (Hafner et al., 2019; Wu et al., 2022) 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 (Lipman et al., 2022) generate actions by learning a continuous-time velocity field that transports samples from a simple prior to an action distribution. In flow-matching VLAs, the VLM backbone first encodes the current observation context into a conditioning representation $\mathcal{K}$. Using $\mathcal{K}$, the action expert 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 $s$. With $N$ Euler steps and $s \in \{0, \frac{1}{N}, \ldots, \frac{N-1}{N}\}$, 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 $\mathbf{x}_1$.

**Contrastive learning and energy-based models.** Contrastive learning trains a scoring function to assign higher compatibility to positive samples than to negative ones. Energy-based models (EBMs) (LeCun et al., 2006) provide a natural formulation for this comparison by assigning each input $\mathbf{z}$ a scalar energy $E_\theta(\mathbf{z}) \in \mathbb{R}$, where lower energy indicates higher compatibility. Given context $\mathbf{c}$, the energy is written as $E_\theta(\mathbf{z} \mid \mathbf{c})$. One common objective for learning such relative preferences is the margin-based ranking loss (Hadsell et al., 2006):

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

where $\mathbf{z}^+$ and $\mathbf{z}^-$ denote positive and negative samples under the same context $\mathbf{c}$, $m > 0$ is the margin, and $[\cdot]_+ = \max(0, \cdot)$. The loss encourages positives to have lower energy than negatives by at least $m$. We therefore formulate action verification as contrastive energy modeling, leveraging the fact that candidate selection requires only relative ranking rather than calibrated absolute rewards.

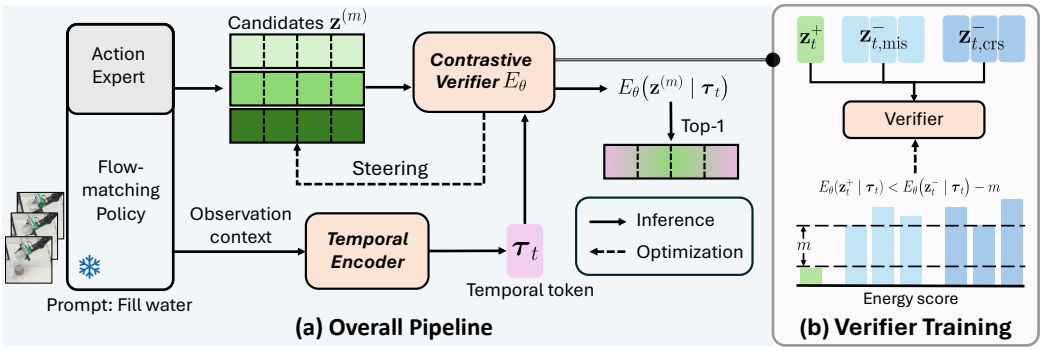


Figure 2: **Illustration of Temporal Verification. (a)** Overall pipeline. We encode observation-action history into a temporal token and use it to condition the energy-based contrastive verifier. At each flow-matching integration step, candidate representations are scored by the verifier. The resulting energy scores are used to guide 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.

## 4 Methodology

**Overview.** Our goal is to augment a pretrained flow-matching VLA with a lightweight verifier that accounts for cross-chunk temporal coherence without requiring additional expert demonstrations. To this end, we propose Temporal Verification (TeV), illustrated in Figure 2(a). TeV first encodes recent observation–action history into a compact *temporal token*, trained with a dynamics-aware objective to preserve execution-relevant temporal information. Using this token as temporal context, we train a lightweight energy-based verifier contrastively (Figure 2(b)). At inference time, the learned energy function provides gradient-based guidance that steers intermediate flow samples toward lower-energy regions during generation, and then ranks the resulting action candidates for execution.

### 4.1 Temporal Token Learning

Augmenting stateless flow-matching VLAs with dense observation histories can incur substantial computational overhead (Torne et al., 2026). We instead compress recent history into a compact *temporal token*, providing the verifier with temporal context at low inference overhead. Given a history window of length $W$, we first construct a compact representation for each timestep $i \in \{t-W+1, \ldots, t\}$. For each observation $\mathbf{o}_i$, we extract its token representations from the pretrained VLM backbone. We project these tokens into a $d_h$-dimensional latent space and append a learnable readout token shared across timesteps. A lightweight transformer processes the resulting sequence with bidirectional attention, and its readout output forms the compressed observation summary $\mathbf{h}_i \in \mathbb{R}^{d_h}$. These representations compactly encode individual observations but remain temporally independent, and therefore do not yet capture how the trajectory evolves over time.

To incorporate execution history, we pair each observation summary $\mathbf{h}_i$ with the immediately preceding executed action $\mathbf{a}_{i-1}$. We concatenate each pair and process the resulting sequence with a causal temporal transformer $\mathcal{T}_\psi$ (Vaswani et al., 2023; Chen et al., 2021):

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

The final output $\boldsymbol{\tau}_t$, termed the **temporal token**, causally summarizes recent observation–action history and provides temporal context for verifying candidate chunks at time $t$.

To encourage the temporal token to capture coherent observation–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 the next $F$ actions. We define

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

where $[\mathbf{a}_t, \ldots, \mathbf{a}_{t+F-1}]$ denotes the ground-truth future action sequence from the offline trajectory. This auxiliary objective serves as a dynamics-aware regularizer, encouraging $\boldsymbol{\tau}_t$ to retain temporally

predictive information that connects recent observations and executed actions to subsequent motion. The resulting representation is thus better suited to temporally aware candidate ranking and steering.

### 4.2 Contrastive Verifier Training

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

For each training timestep $t$, let $\mathbf{z}_t^+$ denote the latent representation of the corresponding expert chunk, which serves as the positive sample using supervision already available in the offline imitation data. We then construct two classes of negatives, each targeting a distinct failure mode of generative action prediction. Contrasting the expert chunk against these negatives trains the verifier to assess the action quality and temporal compatibility required for candidate ranking and generation-time guidance.

❶ **Temporal-mismatch negatives.** Flow-matching policies can model multimodal action distributions, but this flexibility may cause switching between individually plausible yet mutually inconsistent action modes across consecutive chunks, leading to oscillatory behavior and task failure. 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 each positive chunk $\mathbf{z}_t^+$, we sample expert chunks from other timesteps $j$ in the same episode subject to a minimum temporal distance $|j-t| \geq \delta$. Although these chunks originate from expert demonstrations and are therefore plausible in isolation, they are inconsistent with the execution history encoded by $\boldsymbol{\tau}_t$. Contrasting them with $\mathbf{z}_t^+$ encourages the verifier to distinguish the appropriate continuation of the current trajectory from plausible but temporally mismatched alternatives. This construction is particularly useful under state ambiguity, where similar observations may correspond to different stages of execution and therefore require different subsequent actions.

❷ **Coarse-integration negatives.** Temporal-mismatch negatives capture expert-like chunks that are incompatible with the current history, but do not expose the verifier to quality degradation 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 under the same observation context using fewer Euler steps $n < N$. Reducing the number of integration steps yields a coarser approximation of the learned flow, providing a controlled source of imperfect policy-generated chunks. We sample negatives across different values of $n$, with smaller $n$ generally producing easier negatives and larger $n$ producing harder negatives that more closely resemble full-step policy outputs. Contrasting against these negatives encourages the verifier to distinguish high-quality action chunks from suboptimal policy-generated alternatives.

Having defined the negative families, we train the verifier using the margin-based contrastive objective in Eq. 2. Let $\mathcal{R} \coloneqq \{\mathrm{mis}, \mathrm{crs}\}$ denote the set of negative types. For each $r \in \mathcal{R}$, let $\{\mathbf{z}^-_{t,r,k}\}_{k=1}^{K_r}$ denote the corresponding negatives and and $m_r > 0$ its margin. The contrastive verifier loss is

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

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 $m_r$. Together, the two negative families provide complementary supervision for learning both temporal compatibility and action quality required for test-time verification.

Finally, we jointly optimize the temporal encoder and energy-based verifier with

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

where $\lambda_\tau$ controls the contribution of the auxiliary future-action prediction objective.

### 4.3 Verification as Guidance

The learned verifier defines a differentiable energy function over action representations conditioned on recent execution history. This allows the learned energy to guide flow integration in addition to

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

| Method | Camera | Robot | Language | Light | Background | Noise | Layout | Average |
|---|---|---|---|---|---|---|---|---|
| Base | 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 |
| V-GPS | 39.9 | 67.4 | 77.8 | 74.5 | 77.2 | 68.8 | 73.1 | 67.2 |
| LPB | 33.9 | 56.5 | 63.7 | 64.9 | 58.5 | 62.5 | 68.0 | 58.5 |
| DynaGuide | 33.9 | 59.5 | 61.9 | 65.0 | 61.2 | 58.8 | 70.2 | 58.6 |
| TE | 44.4 | 72.3 | 82.2 | 76.3 | 85.1 | 74.8 | 84.0 | 73.0 |
| BID | 44.2 | 73.3 | 76.2 | 81.4 | 80.3 | 76.6 | 81.7 | 72.2 |
| TACO | 44.9 | **77.1** | 90.1 | 77.4 | 86.2 | 77.1 | 87.2 | 76.0 |
| TACO* | 42.7 | 74.8 | 83.8 | 76.6 | 76.8 | 72.2 | 80.1 | 71.5 |
| FOREWARN | 43.4 | 73.3 | 87.5 | 80.7 | 83.0 | 78.6 | 86.2 | 75.0 |
| **Ours** | **48.4** | 75.6 | **91.9** | **84.7** | **95.8** | **80.0** | **92.6** | **79.8** |

ranking completed candidates (Dhariwal & Nichol, 2021). Let $\mathbf{z}_s$ denote the differentiable action representation extracted from the intermediate sample $\mathbf{x}_s$ at flow time $s$. Because $\mathbf{z}_s$ depends on $\mathbf{x}_s$, the energy gradient $\nabla_{\mathbf{x}_s} E_\theta(\mathbf{z}_s \mid \boldsymbol{\tau}_t)$ can be backpropagated through the action expert to guide generation. For each candidate $m$, we sample $\mathbf{x}_0^{(m)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, while the VLA context $\mathcal{K}_t$ is computed once from the current observation and shared across candidates. We then augment the Euler update in Eq. 1 with an energy-gradient term:

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

where $\eta_s \geq 0$ controls the guidance strength at flow time $s$. We linearly decay the $\eta_s$ from $\eta_0 = \eta$ to zero over the integration process, applying stronger guidance early in the flow and progressively recovering the unguided base dynamics near completion. This schedule allows the verifier to influence coarse action structure early while preserving the base policy's fine-grained refinement in later steps.

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

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

and execute $\mathbf{x}_1^{(m^*)}$. The verifier thus serves a dual role, guiding intermediate samples toward lower-energy regions during integration and ranking 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 (Fei et al., 2025) and RoboTwin 2.0 (Chen et al., 2025a). LIBERO-Plus spans 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`). We focus on the most challenging LIBERO10 suite, comprising 2,519 task instances. 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 training and evaluate the policy over 100 independent trials.

**Baselines.** We compare TeV with verification-based and temporal-consistency baselines using the same $\pi_{0.5}$ (Intelligence et al., 2025) backbone and checkpoint. Unless otherwise noted, all test-time scaling methods sample $M = 4$ action chunks per observation.

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 | 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 | 87.0 | 43.0 | 86.0 | 88.0 | 82.0 | 74.0 | 47.0 | 72.4 |
| BID | 87.0 | 59.0 | 87.0 | 91.0 | 76.0 | 77.0 | 47.0 | 74.9 |
| TACO | 81.0 | 46.0 | 87.0 | 85.0 | 80.0 | 62.0 | 49.0 | 70.0 |
| TACO* | 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** |

- **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)** (Zhao et al., 2023): 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)** (Liu et al., 2025): BID samples multiple candidate chunks and selects the one most similar with the previously executed chunk over overlapping timesteps.
- **TACO** (Yang et al., 2025): 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.
- **Other test-time improvement methods.** We additionally compare against broader test-time improvement approaches, including the value-based action reranking method V-GPS (Nakamoto et al., 2025), generative policy-steering methods LPB (Sun & Song, 2025) and DynaGuide (Du & Song, 2025), and the dynamics-model-based verifier FOREWARN (Wu et al., 2025).

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 verification-only, guidance-only, and the full method across different guidance strengths $\eta$. Verification-only consistently improves task success, whereas guidance alone can substantially degrade performance, indicating that gradient-based steering is unreliable without final candidate selection. The benefit of guidance is benchmark dependent. On RoboTwin2.0, adding guidance to verification improves success from 72.7% to 75.1–77.3% across $\eta \in [0.2, 2.0]$, while on LIBERO-Plus it provides little additional gain

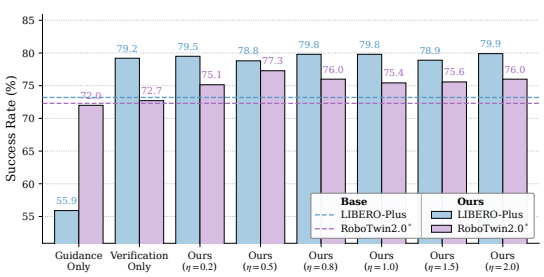


Figure 3: **Effect of guidance and verification.**

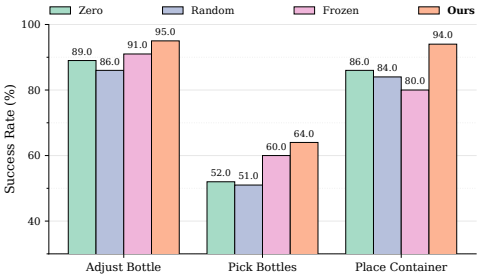


Figure 4: **Effect of temporal token variants.**

over verification alone. We attribute this difference to their distinct roles: verification selects among sampled candidates, whereas guidance refines intermediate samples during flow integration. Guidance is therefore most useful when the initial candidate set lacks a sufficiently accurate action, while final verification mitigates unfavorable steering. Overall, verification provides the robust core improvement, with guidance offering additional gains when candidate refinement is beneficial.

- **Effect of negatives.** We ablate the two negative families on RoboTwin2.0. Starting from a 72.3% base-policy success rate, using only coarse-integration negatives improves performance to 78.1%, while temporal-mismatch negatives alone reach 77.4%. Combining both further increases success to 81.3%, indicating that the two negative families provide complementary training signals.
- **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 $\mathbf{0}$; ***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 (Liu et al., 2025), and TACO (Yang et al., 2025). For execution safety, all methods adopt Temporal Ensembling (Zhao et al., 2023). 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 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.

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

| Method | *Bottle to Bag* | *Cup Transport* | *Water Filling* |
|---|---|---|---|
| Base | 60.7 ± 31.9 | 79.8 ± 27.5 | 40.5 ± 32.4 |
| BID | 59.8 ± 27.8 | 71.5 ± 25.9 | 51.3 ± 30.5 |
| TACO | 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 |

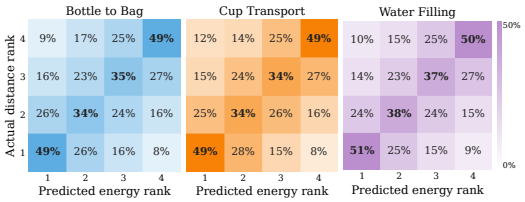


Figure 5: **Energy score reliability.**

**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.

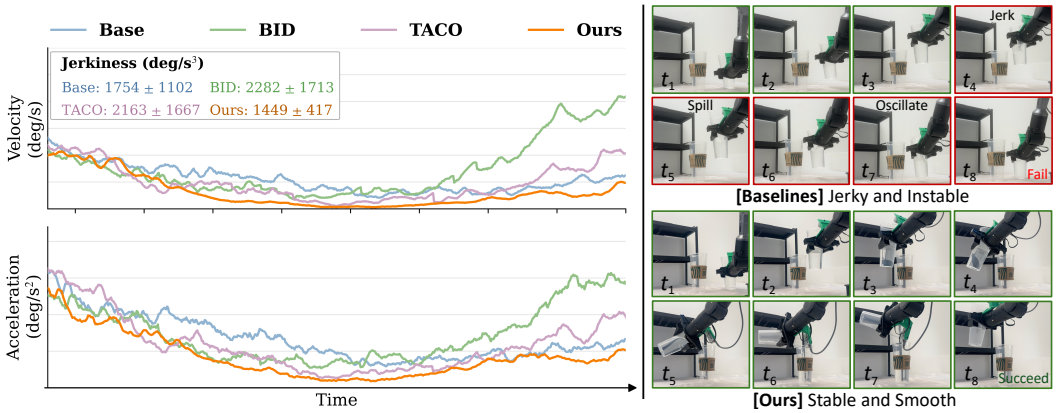


Figure 6: **Illustration of robot kinematics and task progress over time. Left:** Average joint velocity and acceleration during interaction with the water cup. **Right:** Real-world rollouts. The baselines often exhibit abrupt motions and oscillations, leading to water spilling or task failure, whereas 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.

### AI USE STATEMENT

In this work, we used generative AI tools for providing feedback on research methodology and assistance in implementing baselines with unreleased code. We have not used generative AI tools for generating synthetic data sets, helping develop theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assistance in the writing of proofs, proposing or refining hypotheses, designing research methodology or experiments, assisting with translation, cleaning and reformatting dataset, supporting qualitative and thematic data analysis, and interpreting results are not applicable to this work. Additionally, we used generative AI tools for sourcing/searching for information, and providing feedback for the paper to improve readability. We have reviewed all AI-assisted work. We check the AI feedback thoroughly, and verified its code carefully. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

### ETHICS STATEMENT

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.

### REPRODUCIBILITY STATEMENT

To ensure reproducibility, we clarify the training protocols, evaluation metrics, inference settings, and provide a detailed algorithmic description in the Appendix. Hyperparameters, model configurations, and training schedules are reported to allow faithful replication of our results. In addition, source code and pretrained model checkpoints will be released upon request to further support verification and reuse by the community.

## REFERENCES

Yusuf Ali, Gryphon Patlin, Karthik Kothuri, Muhammad Zubair Irshad, Wuwei Liang, and Zsolt Kira. Eve: A generator-verifier system for generative policies, 2025. URL https://arxiv.org/abs/2512.21430.

Kevin Black, Manuel Y. Galliker, and Sergey Levine. Real-time execution of action chunking flow policies, 2025. URL https://arxiv.org/abs/2506.07339.

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. $\pi_0$: A vision-language-action flow model for general robot control, 2026. URL https://arxiv.org/abs/2410.24164.

Haoyuan Cai, Zhenghao Peng, and Bolei Zhou. Predictive preference learning from human interventions, 2025. URL https://arxiv.org/abs/2510.01545.

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. URL https://arxiv.org/abs/2106.01345.

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, 2025a. URL https://arxiv.org/abs/2506.18088.

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, 2025b. URL https://arxiv.org/abs/2510.13778.

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. URL https://arxiv.org/abs/2303.04137.

Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis, 2021. URL https://arxiv.org/abs/2105.05233.

Maximilian Du and Shuran Song. Dynaguide: Steering diffusion polices with active dynamic guidance, 2025. URL https://arxiv.org/abs/2506.13922.

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. URL https://arxiv.org/abs/2510.13626.

Raia Hadsell, Sumit Chopra, and Yann Lecun. Dimensionality reduction by learning an invariant mapping. pp. 1735 – 1742, 02 2006. ISBN 0-7695-2597-0. doi: 10.1109/CVPR.2006.100.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels, 2019. URL https://arxiv.org/abs/1811.04551.

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. URL https://arxiv.org/abs/2504.16054.

Wenqi Jiang, Jason Clemons, Karu Sankaralingam, and Christos Kozyrakis. How fast can i run my vla? demystifying vla inference performance with vla-perf, 2026. URL https://arxiv.org/abs/2602.18397.

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. URL https://arxiv.org/abs/2510.01711.

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. URL https://arxiv.org/abs/2506.17811.

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. URL https://arxiv.org/abs/2602.12281.

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

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. URL https://arxiv.org/abs/2509.00271.

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

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. URL https://arxiv.org/abs/2408.17355.

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. URL https://arxiv.org/abs/2510.09976.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks, 2018. URL https://arxiv.org/abs/1802.05957.

Mitsuhiko Nakamoto, Oier Mees, Aviral Kumar, and Sergey Levine. Steering your generalists: Improving robotic foundation models via value guidance, 2025. URL https://arxiv.org/abs/2410.13816.

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. doi: 10.15607/rss.2025.xxi.113. URL http://dx.doi.org/10.15607/RSS.2025.XXI.113.

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. URL https://arxiv.org/abs/2503.14734.

Zhenghao Peng, Wenjie Mo, Chenda Duan, Quanyi Li, and Bolei Zhou. Learning from active human involvement through proxy value propagation, 2025. URL https://arxiv.org/abs/2502.03369.

Stephane Ross, Geoffrey J. Gordon, and J. Andrew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning, 2011. URL https://arxiv.org/abs/1011.0686.

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. URL https://arxiv.org/abs/2506.01844.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters, 2024. URL https://arxiv.org/abs/2408.03314.

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. URL https://arxiv.org/abs/2603.10052.

Zhanyi Sun and Shuran Song. Latent policy barrier: Learning robust visuomotor policies by staying in-distribution, 2025. URL https://arxiv.org/abs/2508.05941.

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. URL https://arxiv.org/abs/2512.01031.

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. URL https://arxiv.org/abs/2603.03596.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding, 2019. URL https://arxiv.org/abs/1807.03748.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. URL https://arxiv.org/abs/1706.03762.

Haoxuan Wang, Gengyu Zhang, Yan Yan, Yuzhang Shang, Ramana Rao Kompella, and Gaowen Liu. Real-time robot execution with masked action chunking, 2026. URL https://arxiv.org/abs/2601.20130.

Philipp Wu, Alejandro Escontrela, Danijar Hafner, Ken Goldberg, and Pieter Abbeel. Daydreamer: World models for physical robot learning, 2022. URL https://arxiv.org/abs/2206.14176.

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. URL https://arxiv.org/abs/2601.18692.

Yilin Wu, Ran Tian, Gokul Swamy, and Andrea Bajcsy. From foresight to forethought: Vlm-in-the-loop policy steering via latent alignment, 2025. URL https://arxiv.org/abs/2502.01828.

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. URL https://arxiv.org/abs/2511.00091.

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. URL https://www.pi.website/download/rlt.pdf.

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. URL https://arxiv.org/abs/2512.02834.

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. $\chi_0$: Resource-aware robust manipulation via taming distributional inconsistencies, 2026. URL https://arxiv.org/abs/2602.09021.

Hongyin Zhang, Shuo Zhang, Junxi Jin, Qixin Zeng, Runze Li, and Donglin Wang. Robustvla: Robustness-aware reinforcement post-training for vision-language-action models, 2025a. URL https://arxiv.org/abs/2511.01331.

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, 2025b. URL https://arxiv.org/abs/2512.01715.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware, 2023. URL https://arxiv.org/abs/2304.13705.

# APPENDIX

## 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 (Miyato et al., 2018) 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 $\tau_t$. 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.s

Following (Shukor et al., 2025), 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, where progressive pruning is only used when the method supports candidate scoring during generation. Under the same RTX 5080, compilation and applicable cache optimizations, the measured latency ablation for TeV is shown in Figure 5. Our real-world controller operates at 15 Hz, corresponding to 66.7 ms per action step. Using the discretized delay of $\lceil L/66.7 \rceil$, both the base and Verification-Only configurations incur a two-step control delay, whereas the guided configuration incurs a three-step delay. Thus, Verification-Only increases wall-clock latency but remains within the same discrete control-delay bucket as the base policy, while guidance adds one additional control step.

Table 5: **Real-world latency analysis.**

| **Method** | ***Bottle to Bag*** | ***Cup Transport*** | ***Water Filling*** | **Latency $L$ (ms)** |
|---|---|---|---|---|
| Base | $60.7 \pm 31.9$ | $79.8 \pm 27.5$ | $40.5 \pm 32.4$ | $94.8 \pm 0.1$ |
| Base + Verification | $67.2 \pm 28.1$ | $87.2 \pm 20.6$ | $69.8 \pm 14.2$ | $118.9 \pm 0.6$ |
| Base + Verification + Guidance | $69.3 \pm 26.7$ | $88.3 \pm 15.9$ | $68.0 \pm 12.8$ | $159.9 \pm 1.5$ |

**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.



Figure 7: **Robot platform setup.**

**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).
- *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.

**Algorithm 1** Temporal Verification

**Require:** Velocity field $\mathbf{v}_\pi$, verifier $E_\theta$, 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** $\mathbf{x}_1^{(m^*)}$

## D Additional Results

**Contrastive Loss Comparison** Table 6 compares different contrastive objectives for verifier training. In addition to the margin-based contrastive loss used in the main experiments, we also evaluate InfoNCE (van den Oord et al., 2019). Although InfoNCE performs worse than the margin-based objective, it is used without hyperparameter tuning and still yields substantial gains over the 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.

Table 6: **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** |

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

**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 7 reports temporal consistency across methods on the *Adjust Bottle* task. We evaluate consistency for arm pose changes, excluding the gripper, and for the full robot action. Following (Liu et al., 2025), 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

Table 7: **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** |

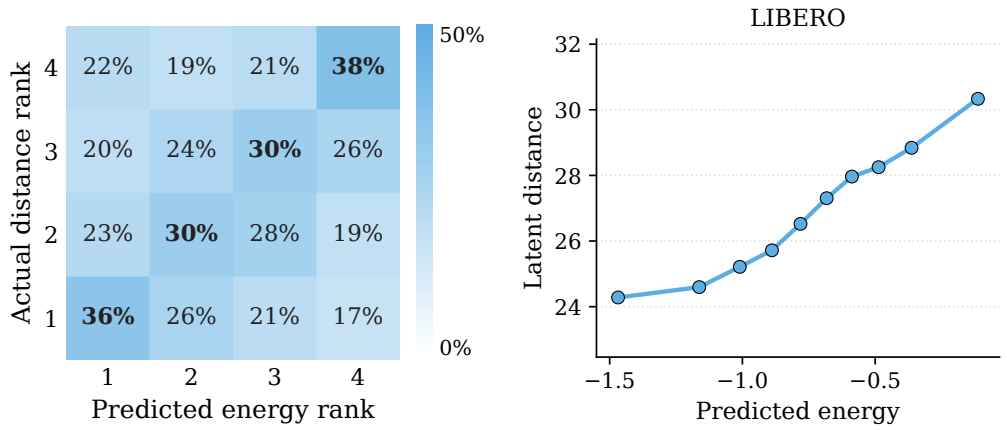


Figure 8: **Energy reliability analysis on LIBERO-Plus. Left:** Rank correlation. **Right:** Energy v.s. Latent Distance.

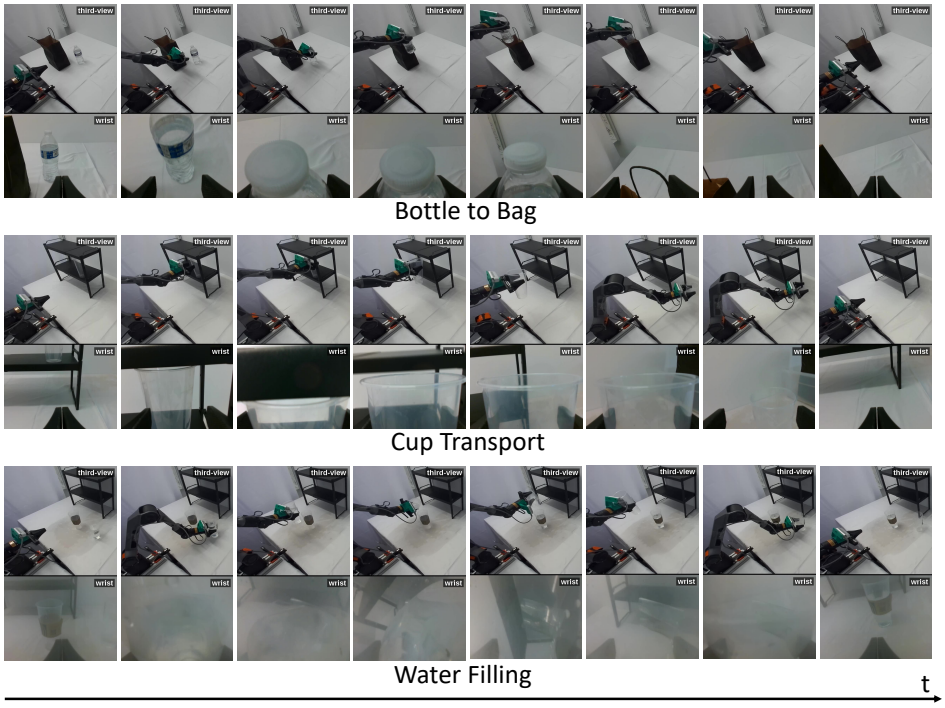


Figure 9: **Real-world task rollouts.**

over $H - 1$ timesteps. The overlap distance is computed as the $\ell_2$ 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.