STABLE SELF-EVOLVING WORLD ACTION MODELS VIA SELF-PERPLEXITY AWARE LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

While self-evolution shows great promise for improving model performance without human demonstrations, how to self-evolving World Action Models (WAMs) that jointly model actions and videos remains unexplored. However, directly fine-tuning WAMs on self-generated trajectories induces a pronounced V-shaped curve where performance drops sharply during early training and then gradually recovers. This may stems from the distribution shift between self-generated trajectories and pre-training human demonstrations. To address this challenge, we propose SPASE, a multi-round self-evolution framework for WAMs. SPASE first introduces an Execution-Grounded Dual Verification module to filter out low-quality data with incorrect actions or inconsistent videos. To alleviate the adverse effects of distribution shift, SPASE introduces Self-Perplexity to measure data divergence from the model's parametric knowledge. Using Self-Perplexity, SPASE adaptively downweights highly divergent data to stabilize training and imposes a deterioration penalty on previously learned data to mitigate catastrophic forgetting. Extensive evaluations across robotic manipulation benchmarks demonstrate the superiority of SPASE. Our code is available at `https://anonymous.4open.science/r/stable_wam_evolution/`.

1 INTRODUCTION

World Action Models (WAMs) unify future world prediction and robot control within a single model, allowing future-state prediction to support policy learning (Wang et al., 2026; Shen et al., 2026a). This joint modeling helps WAMs capture the physical consequences of actions, supporting effective control in complex manipulation tasks (Agarwal et al., 2026; Cen et al., 2025). However, training WAMs relies on large amounts of costly human demonstrations, leading to limited scalability (Ye et al., 2026). Recent research on large language models reveals a promising direction, **self-evolution**, in which iterative training on self-generated data without human annotation can significantly improve model performance (Tao et al., 2024; He et al., 2025).

Few studies have explored self-evolution in Vision-Language-Action (VLA) models and world models (Feng et al., 2026). For VLAs, PLD distills residual-RL recovery trajectories into VLA policies (Xiao et al., 2026), while EVOLVE-VLA adapts models using learned progress feedback (Bai et al., 2025). For world models, WAV uses forward–inverse consistency for self-verification (Liu et al., 2026a), whereas SILVR refines a video planner with feedback-filtered online trajectories (Luo et al., 2026). VLAW further explores co-improvement of VLAs and world models through alternating updates guided by mutual feedback and verification (Guo et al., 2026).

However, a key question remains: *How to self-evolving a WAM by jointly learning from its own actions and videos?* The nucleus of self-evolution lies in leveraging the model's own outputs as training data to improve capabilities, making self-generated data quality the cornerstone of successful evolution. However, different from VLA and world models that solely focus on actions or videos, WAMs are inherently required to model both action and video objectives while capturing their internal relationships, which results in a more entangled and difficult optimization space. Therefore, even a well-trained WAM may still generate incorrect actions (e.g., unstable grasping causing objects to fall) or videos (e.g., inconsistencies with actual execution), as illustrated in Fig.1(c). Directly training on such noisy self-generated data leads to error accumulation over self-evolution. More importantly, even after filtering out these low-quality samples, training on the retained correct data still exhibits

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.

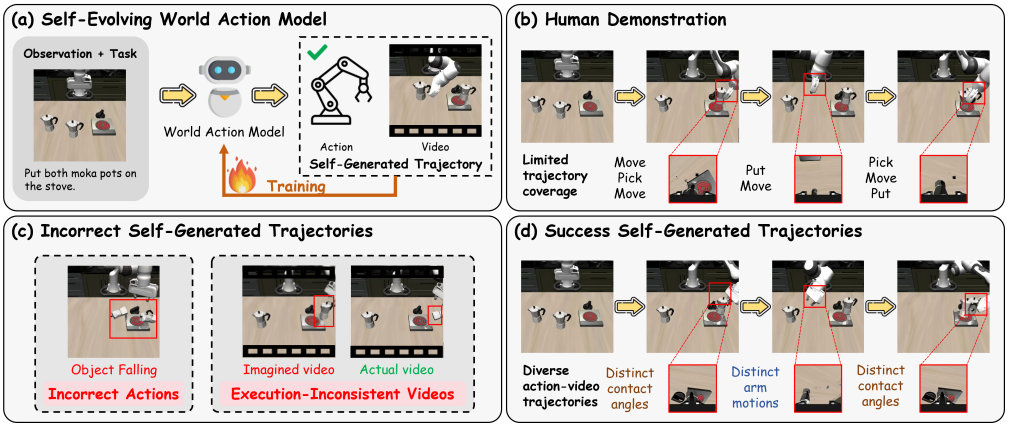


Figure 1: The paradigm of self-evolving WAMs (a), comparing human demonstrations (b) with incorrect self-generated trajectories (c) and success self-generated trajectories (d).

a pronounced *V-shaped curve*: performance drops sharply during early training and then gradually recovers as shown in Fig. 2. This V-shaped curve may stem from the distribution shift between self-generated data and the human demonstrations used for pretraining (Ross et al., 2011; Xiao et al., 2026). For example, Fig. 1(b) depicts a human demonstration completing the task, while Fig. 1(d) shows a successful self-generated trajectory with distinct arm motions and contact angles under the same setting. Such divergence arises because the same task can be completed through diverse action-video trajectories, which human demonstrations cannot fully cover. However, WAM acquires its initial parametric knowledge from these limited coverage demonstrations, directly training on self-generated data with distribution shifts may interfere with existing knowledge and destabilize training (Hancock et al., 2026).

To enable stable self-evolution, a rigorous filtering mechanism is required to exclude low-quality data containing incorrect actions or execution-inconsistent videos. In addition, to alleviate the adverse effects of distribution shift, the self-evolution process of WAMs needs to explicitly incorporate the divergence between self-generated data and the parametric knowledge acquired from human demonstrations. A larger divergence indicates that the data are less consistent with the model's existing parametric knowledge, and directly optimizing on such data may lead to radical parameter updates and destabilize training.

Building on these insights, we propose SPASE, a Multi-round Self-Perplexity Aware self-evolution framework for World Action Models. In each self-evolution round, the WAM trained on previously data serves as the current model with its parametric knowledge. Based on this model, new trajectories are first collected via environment rollouts. Then, an Execution-Grounded Dual Verification module filters self-generated data by first retaining correctly executed trajectories and then removing action–video pairs whose generated videos are inconsistent with actual execution. To alleviate the influence of distribution shift, we introduce **Self-Perplexity**, a model-based proxy that measures the divergence between self-generated data and the WAM's parametric knowledge. Higher values suggesting that training on such data may lead to radical parameter update. SPASE therefore down-weights these samples according to how far their Self-Perplexity exceeds the reference level. In addition, to discourage increases in the previously low Self-Perplexity samples during self-evolution, SPASE penalizes such increases beyond a tolerated margin relative to the historical baseline. Our main contributions are summarized as follows:

- As far as we know, we are the first to study joint action–video self-evolution in WAMs by jointly learning from self-generated actions and videos. And we identify a pronounced V-shaped performance curve that may stem from distribution shift between self-generated trajectories and the human demonstrations.
- We propose SPASE, which introduces Self-Perplexity to measure the divergence between the self-generated data and the WAM's parametric knowledge and drive stable self-evolution.
- Comprehensive evaluations across robotic benchmarks and shows the effectiveness of SPASE in improving training stability, mitigating forgetting, and enhancing task performance.

2 PRELIMINARY ANALYSIS

2.1 SELF-PERPLEXITY FOR ASSESSING DISTRIBUTION SHIFT

To alleviate the V-shaped performance degradation associated with distribution shift, we introduce **Self-Perplexity**, which measures the divergence between self-generated data and the current WAM's parametric knowledge. Specifically, let x_i denote the self-generated data with conditioning input C_i. Consider a fixed set of K queries $\mathcal{Q}_i = \{\xi_{i,k}\}_{k=1}^{K}$, where $\xi_{i,k} = (\tau_{i,k}, \epsilon_{i,k})$ consists of a flow time $\tau_{i,k} \sim \mathcal{U}(0,1)$ and Gaussian noise $\epsilon_{i,k} \sim \mathcal{N}(0, I)$. The Self-Perplexity score is computed by averaging the velocity prediction error over these queries:

$$\bar{\ell}_i(\theta) = \frac{1}{K}\sum_{k=1}^{K} \ell_i(\theta; \xi_{i,k}) = \frac{1}{K}\sum_{k=1}^{K} \left\| v_\theta(x_{\tau_{i,k}}, \tau_{i,k}, C_i) - (\epsilon_{i,k} - x_i) \right\|_2^2 . \tag{1}$$

Here, $x_{\tau_{i,k}} = (1 - \tau_{i,k})x_i + \tau_{i,k}\epsilon_{i,k}$ is the noisy target, and v_θ denotes the conditional velocity field. The query-level error $\ell_i(\theta; \xi_{i,k})$ follows the standard conditional flow matching (CFM) objective for this linear interpolation path (Lipman et al., 2022). Lower Self-Perplexity indicates that the self-generated data are more consistent with the current WAM's parametric knowledge, whereas higher values indicate a larger divergence and direct training on them may require radical parameter updates. In addition, Self-Perplexity can track whether the WAM maintains its previously learned knowledge. For the same data, an increase in Self-Perplexity indicates that the updated WAM has become less consistent with the knowledge previously encoded from these data. This suggests potential forgetting during subsequent training.

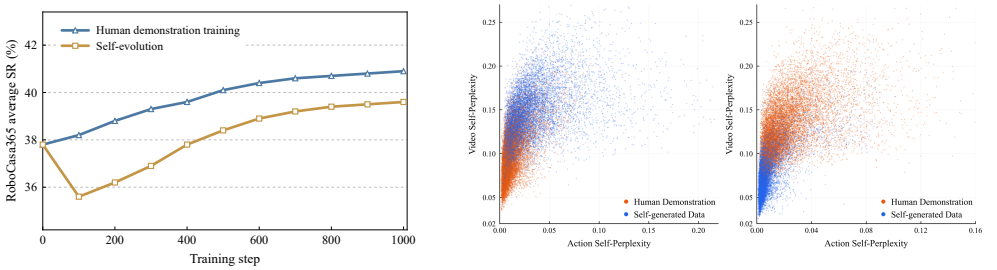
2.2 UNDERSTANDING V-SHAPED CURVE VIA SELF-PERPLEXITY

Our research begins with examining the V-shaped curve through Self-Perplexity in a single round of self-evolution with Cosmos3-Nano (Agarwal et al., 2026) on RoboCasa365 (Nasiriany et al., 2026). Specifically, the WAM is first trained on human demonstrations to establish basic task performance and incorporate the demonstration data into its parametric knowledge. Subsequently, trajectories generated by this WAM are collected in the benchmark environments, with low-quality data discarded using the approach described in Section 3.1. These filtered data are then used for a round of self-evolution over 1k optimization steps, and compared it with continuous training on human demonstrations. After that, model checkpoints are evaluated every 100 optimization steps, and their average success rates are reported. As illustrated in Figure 2(a), continuous training on human demonstrations steadily improves success rates, whereas self-evolution produces a V-shaped curve, with an initial performance drop followed by recovery.

To understand this via Self-Perplexity, human demonstrations and self-generated data from the above training are evaluated across model checkpoints before and after self-evolution. Figures 2(b) and 2(c) plot each training sample's action and video Self-Perplexity on the horizontal and vertical axes, respectively. Before training, self-generated trajectories tend to exhibit higher Self-Perplexity than human demonstrations in Fig. 2(b), indicating a larger divergence from the initial WAM's parametric knowledge. This discrepancy suggests that even successfully executed trajectories can substantially deviate from the knowledge acquired from human demonstrations, reflecting the distribution shift from human demonstrations to self-generated data. After self-evolution, Fig. 2(c) shows that Self-Perplexity decreases on self-generated trajectories but increases on human demonstrations. This indicates that, while the WAM becomes more consistent with the self-generated data, its parametric knowledge simultaneously drifts away from the previously learned human demonstrations, signaling potential forgetting. Together, these observations motivate regulating updates on highly divergent self-generated data while constraining Self-Perplexity increases on previously learned data. Further theoretical analysis is provided in Appendix I.

3 METHODOLOGY

Fig. 3 presents SPASE, a multi-round self-evolution framework for WAMs. In each round, Execution-Grounded Dual Verification first discards data with execution failures or video inconsistent with actual execution. SPASE then uses Self-Perplexity to limit updates on data that diverge substantially



(a) performance in one round self-evolution (b) Self-Perplexity before self-evolution (c) Self-Perplexity after self-evolution

Figure 2: Empirical analysis on RoboCasa365 with Cosmos3-Nano. (a) Success rates during one round of training. (b,c) Self-Perplexity before and after self-evolution.

from the current WAM's parametric knowledge, while constraining Self-Perplexity increases on previously learned data to mitigate forgetting.

3.1 Execution-Grounded Dual Verification

Before self-evolution round r begins, the WAM trained in the previous round, θ_{r-1}, performs environment rollouts to collect self-generated trajectories. For trajectory j, let C_j denote its task instruction and initial observation, and let V_j^{exec} denote the corresponding execution video recorded by the simulator. Each trajectory contains a sequence of samples (C_i, A_i, V_i), where C_i is the input instruction and observation for sample i, and A_i and V_i are the jointly generated actions and imagined video. The verification first identifies the correctly executed prefix of each trajectory and then checks whether the corresponding imagined videos are consistent with actual execution.

Task-grounded subgoal decomposition. Before verifying the trajectory execution, a *Decomposer* receives C_j and decomposes the task into an ordered sequence of subgoals, $\mathcal{S}_j = (\sigma_{j,1}, \sigma_{j,2}, \ldots, \sigma_{j,M_j})$, where M_j denotes the number of subgoals. Each subgoal specifies an observable, task-relevant condition to be achieved. In this way, $\mathcal{S}_j$ specifies what should be accomplished at each stage of the task and serves as a reference for evaluating the trajectory execution.

Execution-grounded progress verification. Given C_j, $\mathcal{S}_j$, and V_j^{exec}, the *Action Verifier* verifies the completion of each subgoal and identifies its corresponding temporal location in the execution video. A subgoal is considered achieved only if its specified condition is satisfied without execution failures such as object drops, failed grasps, or unintended collisions. Instead of assigning a binary success label to the entire trajectory, its longest consecutively valid execution prefix is retained based on verified subgoals. Once a subgoal fails verification, the corresponding segment and all subsequent samples are discarded. This allows correctly executed segments to be preserved even when the complete trajectory eventually fails.

Segment-level video verification. Correct execution does not guarantee that the jointly generated video faithfully predicts the physical consequences of the generated actions. For each retained sample, the *Video Verifier* compares the imagined video V_i with its temporally aligned execution clip V_i^{exec}. The comparison considers two task-relevant aspects: *object state consistency*, which checks whether key objects are in the same states (e.g., whether an object is inside a container); and *interaction consistency*, which verifies whether robot–object interactions match (e.g., whether the robot grasps, contacts, or releases the object). For each aspect, the evaluator returns one of *pass*, *fail*, or *uncertain*. Uncertain cases are re-evaluated using more densely sampled frames for more fine-grained check. A sample is retained only when all aspects are finally judged as *pass*; otherwise, its actions and imagined video are discarded together.

The samples retained after both verification stages form the self-generated dataset $\mathcal{D}_r$ for round r. Each round uses the current dataset and the immediately preceding dataset: $\mathcal{T}_r = \mathcal{D}_r \cup \mathcal{D}_{r-1}$ for self-evolution. For the first round, $\mathcal{D}_0$ denotes a subset selected from the human demonstrations used to pre-train θ_0. More details are provided in Appendix L.

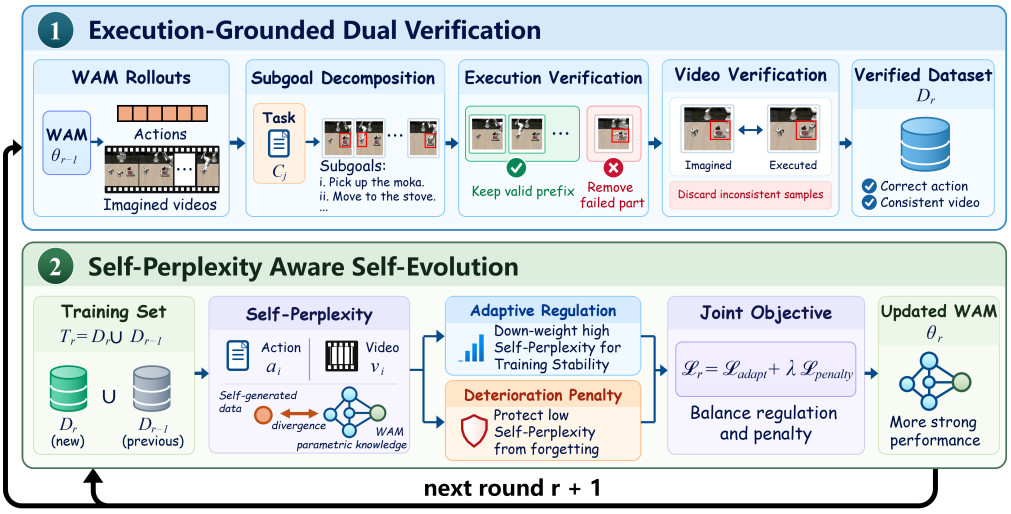


Figure 3: The framework of SPASE, a multi-round self-evolution framework for WAMs.

3.2 Self-Perplexity Aware Self-Evolution

To alleviate training instability and forgetting induced by distribution shift, SPASE uses Self-Perplexity to track how self-generated data diverge from the current WAM's parametric knowledge throughout the self-evolution training process.

Specifically, for each sample i and modality $m \in \{\mathrm{a}, \mathrm{v}\}$, we maintain a learning-state indicator $z_i^m \in \{0, 1\}$, which is periodically updated during training. A relatively high Self-Perplexity indicates that the self-generated data diverge substantially from the current WAM's parametric knowledge. Such data are assigned $z_i^m = 0$ and their updates are constrained. A relatively low Self-Perplexity indicates that the data have been well incorporated into the WAM's parametric knowledge. Such data are assigned $z_i^m = 1$ and should be protected from subsequent increases in Self-Perplexity. During round r, newly collected data in $\mathcal{D}_r$ are initialized with $z_i^m = 0$, while data inherited from $\mathcal{D}_{r-1}$ retain their learning states from the previous round. In the first round, the selected human demonstration in $\mathcal{D}_0$ are initialized with $z_i^m = 1$.

Adaptive Regulation for Training Stability. For data with $z_i^m = 0$, SPASE adjusts their training contribution according to how strongly they diverge from the current WAM's parametric knowledge relative to data regarded as well incorporated into the model. For each modality, a *reference Self-Perplexity level* s_r^m is defined to characterize the typical Self-Perplexity of data that have been well incorporated into the WAM at the beginning of round r:

$$s_r^m = \mathrm{Median}\left(\left\{\bar{\ell}_i^m(\theta_{r,0}) \,\middle|\, i \in \mathcal{D}_{r-1},\ z_i^m = 1\right\}\right) + \eta, \tag{2}$$

where $\theta_{r,0}$ denotes the model at training step $t = 0$, i.e., the beginning of round r, and $\eta > 0$ ensures a positive reference value. The reference Self-Perplexity level s_r^m remains fixed throughout round r to provide a stable reference for measuring divergence gaps and to avoid the criterion drifting with ongoing model updates.

Based on this reference, SPASE periodically evaluates the divergence of each self-generated sample from the current WAM's parametric knowledge, avoiding per-step Self-Perplexity computation. Self-Perplexity is evaluated at the beginning of each round ($t = 0$) and subsequently every H optimization steps. For an arbitrary training step t, let $\hat{t} = H \left\lfloor \frac{t}{H} \right\rfloor$ denote the most recent evaluation step. For each sample under adaptive regulation, its *normalized divergence gap* is computed at $\hat{t}$ as

$$d_{i,r,\hat{t}}^m = \left[\frac{\bar{\ell}_i^m(\theta_{r,\hat{t}})}{s_r^m} - 1 \right]_+, \tag{3}$$

where $[x]_+ = \max(x, 0)$. The gap measures how far the current Self-Perplexity exceeds the reference level. A larger value indicates a greater divergence from the current WAM's parametric knowledge,

while a zero value indicates that the data are within the reference range. Based on this gap, the *adaptive learning weight* is defined as

$$w^m_{i,r,\hat{t}} = \mathrm{sg}\left[\max\left(w_{\min}, \exp\left(-\frac{d^m_{i,r,\hat{t}}}{\tau^m}\right)\right)\right], \tag{4}$$

where sg denotes stop-gradient, $\tau^m > 0$ controls the decay rate, and $0 < w_{\min} \leq 1$ sets the minimum weight. Samples exhibiting larger divergence gaps are assigned smaller learning weights, thereby limiting overly aggressive updates when the data deviate substantially from the current WAM's parametric knowledge. Both $d^m_{i,r,\hat{t}}$ and $w^m_{i,r,\hat{t}}$ are refreshed only at each evaluation step and remain fixed until the next evaluation.

Therefore, for each modality $m \in \{\mathrm{a}, \mathrm{v}\}$, the adaptive-regulation loss at training step t is

$$\mathcal{L}^m_{\mathrm{adapt},i,r,t} = w^m_{i,r,\hat{t}}\ell^m_{i,r,t}, \tag{5}$$

where $\ell^m_{i,r,t}$ denotes the standard CFM training loss evaluated at the current model $\theta_{r,t}$.

Deterioration Penalty for Forgetting Mitigation. At an evaluation step $\hat{t}$, if the Self-Perplexity of a sample under adaptive regulation reaches or falls below the reference Self-Perplexity level,

$$\bar{\ell}^m_i(\theta_{r,\hat{t}}) \leq s^m_r, \tag{6}$$

its learning state is updated as $z^m_i \leftarrow 1$. Under this criterion, the sample is regarded as sufficiently consistent with the WAM's current parametric knowledge and should be protected from subsequent divergence. Meanwhile, the query-level errors underlying its Self-Perplexity at this moment are recorded as a fixed *historical baseline*. Specifically,

$$b^m_i(\xi^m_{i,k}) = \ell^m_i(\theta_{r,\hat{t}}; \xi^m_{i,k}), \qquad k = 1, \ldots, K. \tag{7}$$

This historical baseline remains fixed afterwards and serves as a reference anchor for measuring subsequent deterioration. For each sample with $z^m_i = 1$, SPASE compares the query-level errors $\ell^m_i(\theta_{r,t}; \xi^m_{i,k})$ underlying its Self-Perplexity with its historical baseline $b^m_i(\xi^m_{i,k})$, and penalizes increases beyond a tolerated margin:

$$h^m_{i,r,t} = \frac{1}{K}\sum_{k=1}^{K} \frac{\left[\ell^m_i(\theta_{r,t}; \xi^m_{i,k}) - b^m_i(\xi^m_{i,k}) - \delta^m\kappa^m\right]^2_+}{2\kappa^m}, \tag{8}$$

where $\kappa^m = s^m_1 > 0$ is a fixed scale and $\delta^m \geq 0$ controls the tolerated error increase $\delta^m\kappa^m$. The scale is kept fixed so that the tolerated increase in Self-Perplexity remains consistent across self-evolution rounds. Small variations around the historical baseline are ignored, whereas deterioration beyond the tolerated margin is increasingly penalized. Hence, for each modality $m \in \{\mathrm{a}, \mathrm{v}\}$, the deterioration penalty loss is

$$\mathcal{L}^m_{\mathrm{penalty},i,r,t} = h^m_{i,r,t}. \tag{9}$$

Joint training objective. Each modality contributes its adaptive regulation loss when $z^m_i = 0$ and its deterioration penalty loss when $z^m_i = 1$. The joint objective for each sample is defined as:

$$\begin{aligned}\mathcal{L}_{i,r,t} &= \sum_{m\in\{\mathrm{a},\mathrm{v}\}} \left[(1 - z^m_i)\mathcal{L}^m_{\mathrm{adapt},i,r,t} + \lambda_{\mathrm{stable}} z^m_i \mathcal{L}^m_{\mathrm{penalty},i,r,t}\right] \\ &= (1 - z^{\mathrm{a}}_i)w^{\mathrm{a}}_{i,r,\hat{t}}\ell^{\mathrm{a}}_{i,r,t} + \lambda_{\mathrm{stable}} z^{\mathrm{a}}_i h^{\mathrm{a}}_{i,r,t} + (1 - z^{\mathrm{v}}_i)w^{\mathrm{v}}_{i,r,\hat{t}}\ell^{\mathrm{v}}_{i,r,t} + \lambda_{\mathrm{stable}} z^{\mathrm{v}}_i h^{\mathrm{v}}_{i,r,t}.\end{aligned} \tag{10}$$

where $\lambda_{\mathrm{stable}} \geq 0$ controls the strength of the deterioration penalty. The WAM is optimized using the mini-batch average of Eq. (10) over $\mathcal{T}_r$. Because action and video maintain independent learning states, the two modalities of the same sample may follow different optimization behaviors. For example, the action may already be protected by the deterioration penalty while the corresponding video remains under adaptive regulation. See Algorithm 1 for details and Appendix J for theoretical analyses of Self-Perplexity-based adaptive weighting and the deterioration penalty.

Table 1: Success rates (%) on three benchmarks. **Bold** and underlined denote the best and second-best results within each base model.

Method	RoboCasa365				RoboLab				LIBERO-10
	Atomic	Composite-Seen	Composite-Unseen	**Avg.**	Simple	Moderate	Complex	**Avg.**	**Avg.**
Cosmos3-Nano	67.3	32.8	9.6	37.8	40.8	35.1	25.9	36.8	94.6
+ *Vanilla Self-evolution*	73.2	35.6	11.2	41.3	45.6	38.3	26.8	40.6	96.0
+ *Self-evolution+ER*	78.7	37.2	12.5	44.2	50.6	42.3	28.9	44.8	96.6
+ *LoRA+ER*	80.1	38.5	13.3	45.4	49.1	44.9	30.1	45.0	97.6
+ *L2-SP+ER*	79.5	41.1	13.9	46.2	51.6	43.8	29.8	46.0	97.4
+ *Velocity Regularization+ER*	78.9	40.4	14.4	45.9	51.0	44.2	32.8	46.2	97.8
+ SPASE	**84.7**	**45.8**	**18.6**	**51.1**	**56.4**	**49.8**	**37.3**	**51.5**	**98.6**
DreamZero	62.6	27.2	8.8	34.1	25.8	30.6	13.4	25.6	91.7
+ *Vanilla Self-evolution*	68.3	29.6	9.7	37.2	30.3	34.9	15.1	29.6	91.2
+ *Self-evolution+ER*	73.7	31.2	10.7	39.9	33.9	39.1	16.8	33.2	95.5
+ *LoRA+ER*	74.9	31.9	11.5	40.9	33.5	40.2	17.0	33.3	**97.7**
+ *L2-SP+ER*	75.4	32.8	11.1	41.2	34.0	39.3	17.7	33.4	96.5
+ *Velocity Regularization+ER*	74.2	33.7	12.0	41.3	35.0	39.6	17.2	34.0	97.1
+ SPASE	**79.7**	**38.1**	**16.2**	**46.1**	**39.3**	**44.8**	**22.4**	**38.7**	**97.7**
LingBot-VA	60.8	24.7	7.5	32.2	32.4	29.8	17.6	29.5	92.8
+ *Vanilla Self-evolution*	66.8	27.8	9.5	36.0	37.2	32.5	19.2	33.1	94.5
+ *Self-evolution+ER*	71.6	30.5	10.6	38.9	42.1	37.4	23.2	37.9	95.9
+ *LoRA+ER*	73.4	31.8	11.2	40.2	42.6	40.2	23.8	39.2	97.2
+ *L2-SP+ER*	72.8	33.1	11.7	40.5	44.7	39.5	24.2	40.1	96.8
+ *Velocity Regularization+ER*	72.2	32.6	12.0	40.3	44.3	40.1	25.0	40.2	97.0
+ SPASE	**77.9**	**37.4**	**16.1**	**45.2**	**48.9**	**44.6**	**29.2**	**44.7**	**98.7**

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Benchmark. Evaluation is primarily conducted on RoboCasa365 (Nasiriany et al., 2026), which comprises diverse kitchen manipulation environments across three task splits: Atomic-Seen, Composite-Seen, and Composite-Unseen. Additionally, results on RoboLab (Yang et al., 2026a) under the default language specification (covering simple, moderate, and complex tasks) and LIBERO-10 (Liu et al., 2023) are reported to demonstrate broader generalization.

Metrics. Task success rate (SR) is reported for action evaluation. Video evaluation uses PSNR (Hore & Ziou, 2010), SSIM (Wang et al., 2004), LPIPS (Zhang et al., 2018), FID (Heusel et al., 2017) and FVD (Unterthiner et al., 2018). All simulation evaluations use three different random seeds, with 50 episodes per task per seed, and results are averaged across the three seeds. A qualitative Franka evaluation of Cosmos3-Nano after self-evolution in RoboLab is provided in Appendix H.

Baseline. SPASE is compared with the base model and five training baselines. (1) *Vanilla Self-evolution* applies standard CFM training using only the verified self-generated data each round. (2) *Self-evolution+ER* additionally replays verified data from the previous round each round. (3) *LoRA+ER* updates only low-rank adapters while freezing the original model weights (Hu et al., 2021). (4) *L2-SP+ER* adds an L_2 penalty on parameter deviations from the checkpoint at the start of each round, adapting L2-SP (Xuhong et al., 2018) to multi-round self-evolution. (5) *Velocity Regularization+ER* penalizes the MSE between the current and frozen round-start models' action and video velocity predictions on replay samples (Fan et al., 2025). Baselines (2)–(5) use the same replay setting, while differing in their optimization strategies. More Details are summarized in Appx.K.2.

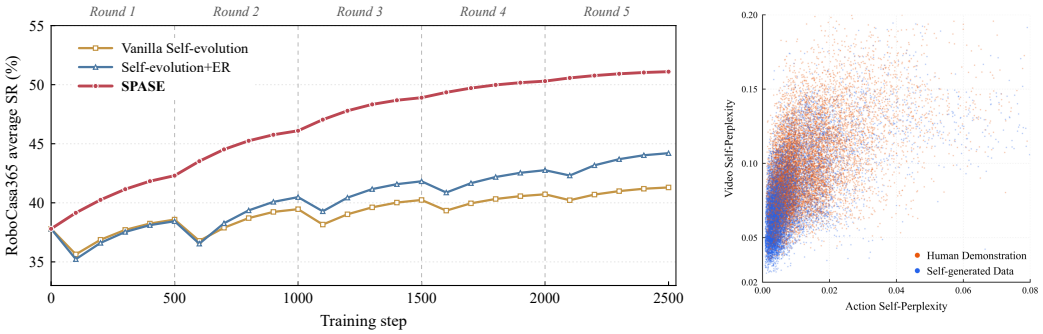
Implementation Details. Cosmos3-Nano (Agarwal et al., 2026), DreamZero (Ye et al., 2026) and LingBot (Li et al., 2026) serve as the base models. Self-evolution runs for 5 rounds, with 500 optimization steps each round. The number of fixed queries is $K = 4$, and the Self-Perplexity evaluation interval is $H = 20$ steps. By default, the hyperparameters are set to temperatures $\tau^{\mathrm{a}} = \tau^{\mathrm{v}} = 1$, tolerances for Self-Perplexity increases $\delta^{\mathrm{a}} = \delta^{\mathrm{v}} = 0.1$, and penalty coefficient $\lambda_{\mathrm{stable}} = 2$. The minimum weight is $w_{\min} = 0.05$, and the numerical stability constant is $\eta = 10^{-6}$. More details are provided in Appx.K.

Table 2: Task success and cross-task generalization.

Method	RoboCasa 365	RoboLab	LIBERO-10	LIBERO-10-to-Goal
Cosmos3-Nano	37.8	36.8	94.6	35.0
SPASE action-only	**53.4**	**53.6**	98.2	19.2
SPASE	51.1	51.5	**98.6**	**37.6**

Table 3: Video quality averaged across three benchmarks.

Method	PSNR ↑	SSIM ↑	LPIPS ↓	FID ↓	FVD ↓
Cosmos3-Nano	23.8	0.822	**0.103**	34.6	106.5
SPASE action-only	20.7	0.767	0.166	38.5	160.4
SPASE	**25.1**	**0.851**	0.109	**33.2**	**99.8**



(a) Performance across 5 rounds.

(b) Self-Perplexity after self-evolution

Figure 4: Stability across five self-evolution rounds with Cosmos3-Nano.

4.2 Main Results

Table 1 shows that SPASE achieves best aggregate performance across all three base models and all three benchmarks, supporting its generalizability across different WAMs and task settings. The gains extend to RoboCasa365's Composite-Unseen tasks, indicating that self-evolution remains beneficial even where the base models have limited initial capabilities. Since all training baselines use verified self-generated data and several additionally constrain model updates, these comparisons suggest the value of Self-Perplexity aware optimization beyond data verification and generic update constraints.

4.3 Stability Across Multi-round Self-Evolution

To assess stability across self-evolution, SPASE is compared with Vanilla Self-evolution and Self-evolution+ER over 5 rounds, with RoboCasa365 average SR evaluated every 100 training steps. Figure 4(a) shows that SPASE improves steadily throughout all 5 rounds, avoiding the recurring V-shaped drops, indicating its stability. Although ER gradually outperforms Vanilla, it still exhibits initial performance drops, suggesting that replay aids recovery but does not resolve the divergency between new data and the model's parametric knowledge induced by distribution shift. Figure 4(b) further shows substantial overlap between the Self-Perplexity distributions of human demonstrations and self-generated data after self-evolution, with less separation than in Figure 2(c). This overlap aligns with more compatible representation of the two data within the model's parametric knowledge; the accompanying stability results suggest mitigation of the adverse effects of distribution shift.

4.4 Importance of Video Supervision

To examine the importance of video supervision, SPASE is compared with the base model and an action-only variant that removes video loss. Tables 2 and 3 show that action-only training achieves higher SR on RoboCasa365 and RoboLab but degrades video quality below the base model. In contrast, SPASE achieves only slightly lower SR while improving most video quality over the base model. These results suggest that action-only training can yield larger gains but degrades video prediction, indicating a degradation of world-modeling capability that may impair out-of-distribution generalization. Consistent with this, direct evaluation of LIBERO-10-trained models on LIBERO-Goal without further adaptation shows that the action-only variant underperforms the base model, whereas SPASE improves upon it (Table 2, last column). These results support joint action–video self-evolution for balancing task improvement, world-modeling capability, and generalization.

Table 4: Ablation study for the proposed SPASE.

Method	RoboCasa365	Δ	RoboLab	Δ	LIBERO-10	Δ
SPASE	**51.1**	–	**51.5**	–	**98.6**	–
Ablations on Execution-Grounded Dual Verification						
⊢ w/o Action Verification	43.7	**−7.4**	42.7	**−8.8**	95.2	**−3.4**
⊢ w/o Video Verification	46.4	−4.7	45.8	−5.7	96.4	−2.2
⊢ Successful Trajectories Only	48.3	−2.8	47.9	−3.6	97.4	−1.2
Ablations on Self-Perplexity Aware Self-Evolution						
⊢ w/o Adaptive Weighting w	45.2	−5.9	44.4	−7.1	96.1	−2.5
⊢ w/o Deterioration Penalty h	44.5	**−6.6**	43.5	**−8.0**	95.6	**−3.0**
⊢ w/o Modality m	47.7	−3.4	47.4	−4.1	97.1	−1.5
⊢ Fixed Reference Level s_r^m	49.2	−1.9	49.2	−2.3	97.7	−0.9

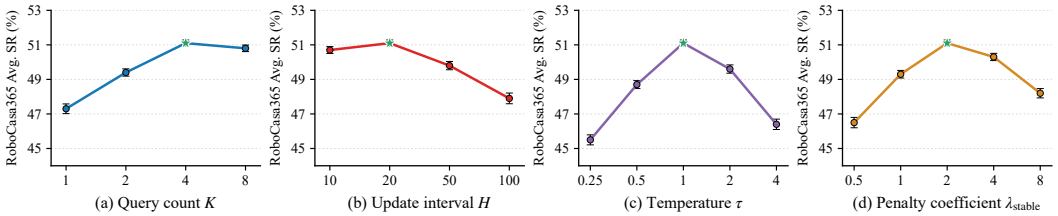


Figure 5: Hyperparameter analysis of the proposed SPASE.

4.5 Ablation Study

Table 4 evaluates the contribution of each component. For execution-grounded dual verification, removing action verification, removing video verification, or retaining only fully successful trajectories reduces performance, highlighting the importance of data quality and the additional learning signals from valid prefixes of failed trajectories. For Self-Perplexity-aware training, removing the adaptive learning weight w lowers SR, suggesting that highly divergent data require conservative updates. Replacing the deterioration penalty h with standard CFM leads to a substantial decline, highlighting the importance of protecting acquired knowledge. Sharing a learning state across actions and videos also degrades performance, suggesting that tthe two modalities benefit from separate protection as they progress at different rates. Fixing the reference Self-Perplexity level to $s_r^m = s_1^m$ yields weaker results, supporting reference levels that track the model's evolving parametric knowledge.

4.6 Hyperparameter Analysis

Fig. 5 examines SPASE's sensitivity to query count K, evaluation interval H, temperature τ, and penalty coefficient λ_{stable}. The best performance is achieved with $K = 4$, $H = 20$, $\tau = 1$, and $\lambda_{\text{stable}} = 2$. For K, too few queries may reduce Self-Perplexity estimation reliability, while more than four add cost without further gains. For H, shorter intervals offer little benefit, whereas longer intervals delay the adjustment of learning weights and states to the model's evolving parametric knowledge. For τ, small values can overly suppress learning from new experiences, while large values weaken regulation on highly divergent data. For λ_{stable}, weak penalties may permit forgetting, whereas overly strong penalties can restrict adaptation to new data.

5 Conclusion

In this work, we propose SPASE to enable stable joint action–video self-evolution in World Action Models. SPASE first applies Execution-Grounded Dual Verification to filter unreliable self-generated data. It then introduces Self-Perplexity to measure the divergence between self-generated data and the WAM's parametric knowledge. SPASE downweights data above the reference Self-Perplexity level to regulate updates, while penalizing score increases on previously learned data to mitigate forgetting. Extensive experiments show that SPASE improves task performance and stabilizes training.

AI USE STATEMENT

Generative AI tools were used to assist with manuscript drafting and language polishing (e.g., translation, LaTeX formatting), and to support literature retrieval and the discovery of related work. All AI-assisted content was reviewed by the authors, who take full responsibility for the final content of this paper.

REFERENCES

Niket Agarwal, Arslan Ali, Jon Allen, Martin Antolini, Adeline Aubame, Alisson Azzolini, Junjie Bai, Maciej Bala, Yogesh Balaji, Josh Bapst, et al. Cosmos 3: Omnimodal world models for physical ai. *arXiv preprint arXiv:2606.02800*, 2026.

Zechen Bai, Chen Gao, and Mike Zheng Shou. Evolve-vla: Test-time training from environment feedback for vision-language-action models. *arXiv preprint arXiv:2512.14666*, 2025.

Jun Cen, Chaohui Yu, Hangjie Yuan, Yuming Jiang, Siteng Huang, Jiayan Guo, Xin Li, Yibing Song, Hao Luo, Fan Wang, et al. Worldvla: Towards autoregressive action world model. *arXiv preprint arXiv:2506.21539*, 2025.

Jiajun Fan, Shuaike Shen, Chaoran Cheng, Yuxin Chen, Chumeng Liang, and Ge Liu. Online reward-weighted fine-tuning of flow matching with wasserstein regularization. In *The Thirteenth International Conference on Learning Representations*, 2025.

Tongtong Feng, Xin Wang, and Wenwu Zhu. Self-evolving embodied ai. *arXiv preprint arXiv:2602.04411*, 2026.

Yanjiang Guo, Jianke Zhang, Xiaoyu Chen, Xiang Ji, Yen-Jen Wang, Yucheng Hu, and Jianyu Chen. Improving vision-language-action model with online reinforcement learning. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 15665–15672. IEEE, 2025.

Yanjiang Guo, Tony Lee, Lucy Xiaoyang Shi, Jianyu Chen, Percy Liang, and Chelsea Finn. Vlaw: Iterative co-improvement of vision-language-action policy and world model. *arXiv preprint arXiv:2602.12063*, 2026.

Asher Hancock, Xindi Wu, Lihan Zha, Olga Russakovsky, and Anirudha Majumdar. Actions as language: Fine-tuning vlms into vlas without catastrophic forgetting. In *International Conference on Learning Representations*, volume 2026, pp. 75333–75360, 2026.

Tao He, Hao Li, Jingchang Chen, Runxuan Liu, Yixin Cao, Lizi Liao, Zihao Zheng, Zheng Chu, Jiafeng Liang, Ming Liu, et al. Breaking the reasoning barrier a survey on llm complex reasoning through the lens of self-evolution. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 7377–7417, 2025.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

Alain Hore and Djemel Ziou. Image quality metrics: Psnr vs. ssim. In *2010 20th international conference on pattern recognition*, pp. 2366–2369. IEEE, 2010.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

Lin Li, Qihang Zhang, Yiming Luo, Shuai Yang, Ruilin Wang, Fei Han, Mingrui Yu, Zelin Gao, Nan Xue, Xing Zhu, et al. Causal world modeling for robot control. *arXiv preprint arXiv:2601.21998*, 2026.

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

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023.

Yuejiang Liu, Fan Feng, Lingjing Kong, Weifeng Lu, Jinzhou Tang, Kun Zhang, Kevin Murphy, Chelsea Finn, and Yilun Du. World action verifier: Self-improving world models via forward-inverse asymmetry. *arXiv preprint arXiv:2604.01985*, 2026a.

Zeting Liu, Zida Yang, Zeyu Zhang, and Hao Tang. Evovla: Self-evolving vision-language-action model. In *European Conference on Computer Vision*, pp. 241–260. Springer, 2026b.

Calvin Luo, Zilai Zeng, Mingxi Jia, Yilun Du, and Chen Sun. Self-improving loops for visual robotic planning. In *International Conference on Learning Representations*, volume 2026, pp. 126885–126913, 2026.

Soroush Nasiriany, Sep Nasiriany, Abhiram Maddukuri, and Yuke Zhu. Robocasa365: A large-scale simulation framework for training and benchmarking generalist robots. In *International Conference on Learning Representations*, volume 2026, pp. 98643–98667, 2026.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 627–635. JMLR Workshop and Conference Proceedings, 2011.

Qiuhong Shen, Shihua Zhang, Yue Liao, Qi Li, Zhenxiong Tan, Shizun Wang, Shuicheng Yan, and Xinchao Wang. World action models: A survey. *arXiv preprint arXiv:2606.20781*, 2026a.

Yichao Shen, Fangyun Wei, Zhiying Du, Yaobo Liang, Yan Lu, Jiaolong Yang, Nanning Zheng, and Baining Guo. Videovla: Video generators can be generalizable robot manipulators. *Advances in neural information processing systems*, 38:95597–95621, 2026b.

Sebastian Stapf, Pablo Acuaviva Huertos, Aram Davtyan, and Paolo Favaro. World model self-distillation: Training world models to solve general tasks. *arXiv preprint arXiv:2606.12072*, 2026.

Zhengwei Tao, Ting-En Lin, Xiancai Chen, Hangyu Li, Yuchuan Wu, Yongbin Li, Zhi Jin, Fei Huang, Dacheng Tao, and Jingren Zhou. A survey on self-evolution of large language models. *arXiv preprint arXiv:2404.14387*, 2024.

Thomas Unterthiner, Sjoerd Van Steenkiste, Karol Kurach, Raphael Marinier, Marcin Michalski, and Sylvain Gelly. Towards accurate generative models of video: A new metric & challenges. *arXiv preprint arXiv:1812.01717*, 2018.

Siyin Wang, Junhao Shi, Zhaoyang Fu, Xinzhe He, Feihong Liu, Chenchen Yang, Yikang Zhou, Zhaoye Fei, Jingjing Gong, Jinlan Fu, et al. World action models: The next frontier in embodied ai. *arXiv preprint arXiv:2605.12090*, 2026.

Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

Wenli Xiao, Haotian Lin, Andy Peng, Haoru Xue, Tairan He, Zhengyi Luo, Yuqi Xie, Fengyuan Hu, Jim Fan, Guanya Shi, et al. Self-improving vision-language-action models with data generation via residual rl. In *International Conference on Learning Representations*, volume 2026, pp. 13209–13236, 2026.

LI Xuhong, Yves Grandvalet, and Franck Davoine. Explicit inductive bias for transfer learning with convolutional networks. In *International conference on machine learning*, pp. 2825–2834. PMLR, 2018.

Jenai Xuning Yang, Rishit Dagli, Alex Zook, Hugo Hadfield, Ankit Goyal, Stan Birchfield, Fabio Ramos, and Jonathan Tremblay. Robolab: A high-fidelity simulation benchmark for analysis of task generalist policies. *arXiv preprint arXiv:2604.09860*, 2026a.

Jiazhi Yang, Kunyang Lin, Jinwei Li, Wencong Zhang, Tianwei Lin, Longyan Wu, Zhizhong Su, Hao Zhao, Ya-Qin Zhang, Li Chen, et al. Rise: Self-improving robot policy with compositional world model. *arXiv preprint arXiv:2602.11075*, 2026b.

Sherry Yang, Yilun Du, Kamyar Ghasemipour, Jonathan Tompson, Leslie Kaelbling, Dale Schuurmans, and Pieter Abbeel. Learning interactive real-world simulators. *arXiv preprint arXiv:2310.06114*, 2023.

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

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *2018 IEEE/CVF conference on computer vision and pattern recognition*, pp. 586–595. IEEE, 2018.

Chuning Zhu, Raymond Yu, Siyuan Feng, Benjamin Burchfiel, Paarth Shah, and Abhishek Gupta. Unified world models: Coupling video and action diffusion for pretraining on large robotic datasets. *arXiv preprint arXiv:2504.02792*, 2025.

A Related Work

World Action Models. World Action Models (WAMs) bridge Vision-Language-Action policies, which map observations and instructions to actions (Kim et al., 2024), and world models, which predict the consequences of interactions (Yang et al., 2023). A prominent approach jointly models actions and visual futures, linking control to predicted outcomes (Shen et al., 2026a). Unified World Models couple action and video diffusion, supporting policy learning, dynamics prediction, and training on action-free videos (Zhu et al., 2025; Shen et al., 2026b). WorldVLA instead unifies action and visual prediction through autoregressive modeling (Cen et al., 2025). DreamZero uses video pretraining for joint action–video generation and demonstrates generalization to unseen tasks and environments (Ye et al., 2026). LingBot-VA integrates frame prediction and action generation with causal modeling and closed-loop feedback (Li et al., 2026). Cosmos 3 jointly models language, images, video, audio, and actions within an omnimodal architecture (Agarwal et al., 2026).

Self-Evolving Embodied AI. Research on self-evolving embodied AI explores autonomous improvement through interaction data and feedback (Feng et al., 2026; Guo et al., 2025; Yang et al., 2026b; Stapf et al., 2026). For VLAs, PLD trains residual policies via reinforcement learning and distills hybrid base-policy and recovery trajectories into the VLA (Xiao et al., 2026). EVOLVE-VLA uses a learned progress estimator to guide test-time training, with feedback smoothing and progressive horizon extension to stabilize learning (Bai et al., 2025). EvoVLA combines stage-aligned rewards, pose-based exploration, and memory for long-horizon manipulation (Liu et al., 2026b). For world models, WAV compares proposed subgoals with forward rollouts of inferred actions to guide exploration (Liu et al., 2026a). SILVR refines a video planner using execution data filtered by task-success or vision-language-model feedback (Luo et al., 2026). VLAW connects these directions: real rollouts improve a world model, which generates additional trajectories for updating a separate VLA policy (Guo et al., 2026). SPASE uses its own verified action and video outputs as supervision, with modality-specific Self-Perplexity regulation and deterioration penalties.

B Self-Perplexity across Learning Histories

Interpretation and scope. Self-Perplexity in Sec. 2.1 measures the divergence between data and the WAM's parametric knowledge. Here, divergence denotes model-relative inconsistency, assessed

through the CFM velocity prediction errors in Eq. (1). The learned conditional velocity field encodes a clean-target predictor, and Self-Perplexity is exactly its time-normalized squared discrepancy from the evaluated target (Lemma 1). At the unrestricted population CFM optimum, this predictor equals the conditional mean of the clean target, connecting the assessment to the predictive structure learned from data. This interpretation concerns consistency with the current model and does not identify Self-Perplexity with a statistical divergence between probability distributions. Appendix I establishes this predictive interpretation, gives a calibration bound relative to a reference knowledge field, and distinguishes changes on fixed data from absolute-score comparisons across data sources.

Experimental setup. To examine how this assessment changes with learning history beyond checkpoint-independent differences between data sources, two Cosmos3-Nano branches start from θ_0 on RoboCasa365 and follow the 1,000-step joint CFM setting in Sec. 2.2 with matched optimization and task sampling. The human-continuation branch θ_{H} trains on human demonstrations, whereas the self-evolution branch θ_{G} trains on verified data generated by θ_0. The initial model and both endpoints are evaluated on fixed sets of 100 human and 100 verified self-generated trajectories with matched task composition. Each trajectory contributes two non-overlapping action–video windows, matched across sources by length and phase; video targets are demonstration frames and model-generated frames, respectively. All evaluation trajectories are excluded from both post-initialization training streams and reference calibration.

Evaluation. Each target uses 16 evaluation queries independent of training and fixed across checkpoints, with identical loss definitions, masks, and normalization. Scores are normalized by the initial reference s_1^m, fixed across checkpoints and sources, then averaged over queries and windows within trajectories, trajectories within tasks, and equally across tasks. Let $S^m_{b,d}$ denote the normalized score for checkpoint $b \in \{0, \mathrm{H}, \mathrm{G}\}$, data source $d \in \{\mathrm{H}, \mathrm{G}\}$, and modality $m \in \{\mathrm{a}, \mathrm{v}\}$. The source interaction is

$$I^m = \left(S^m_{\mathrm{H,G}} - S^m_{\mathrm{G,G}}\right) - \left(S^m_{\mathrm{H,H}} - S^m_{\mathrm{G,H}}\right). \tag{11}$$

Positive I^m indicates a greater relative advantage of self-generated-data training on generated targets than on human targets. This comparison uses checkpoint differences on each fixed data source, following Eq. (23); measured scores and interactions are averaged across three paired training seeds.

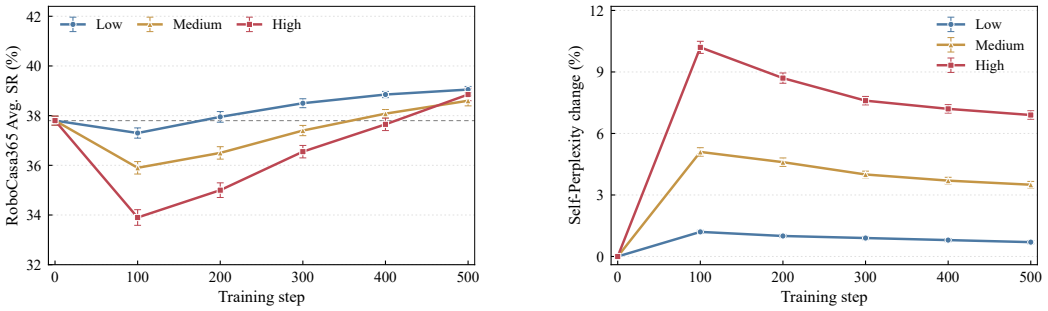
Table 5: Normalized Self-Perplexity across training histories (↓).

Checkpoint	Action		Video	
	Human	Generated	Human	Generated
Initial θ_0	1.04	1.56	1.06	1.47
Human continuation θ_{H}	0.91	1.44	0.95	1.39
Self-evolution θ_{G}	1.20	1.16	1.18	1.14
Source interaction I^m	0.57		0.48	

Results. Table 5 compares how the two training histories affect scores on fixed evaluation data. Human continuation lowers scores on both sources, while the self-evolution branch shows larger reductions on generated targets and increases on human targets, consistent with the opposing trends in Figure 2. The positive source interactions describe a training-source-dependent advantage, beyond a checkpoint-independent score offset between the two datasets. This pattern supports the model-relative interpretation of divergence from parametric knowledge in Sec. 2.1, without establishing a monotonic relationship with task success. Appendices C and D further examine its association with training instability and utility for assigning learning weights.

C Initial Self-Perplexity and Early Training Instability

Experimental setup. To examine whether initial Self-Perplexity indicates training instability, low-, medium-, and high-score subsets are compared using Cosmos3-Nano on RoboCasa365. Training and evaluation follow Section 4.1 and Appendix K, using one 500-step round and checkpoints every 100 steps. All samples are generated by the same initial model θ_0 and pass dual verification. For



(a) Performance during one round.

(b) Self-Perplexity on human data.

Figure 6: One-round training on low-, medium-, and high-Self-Perplexity data. (a) RoboCasa365 SR. (b) Self-Perplexity change on human demonstrations.

grouping, action and video Self-Perplexity are normalized by their reference levels and averaged:

$$\frac{1}{2}\sum_{m\in\{\mathrm{a},\mathrm{v}\}}\frac{\bar{\ell}_i^m(\theta_0)}{s_1^m}. \tag{12}$$

Within each task, samples are stratified by window length, trajectory position, and full-success or valid-prefix provenance, then divided into score tertiles. Equal numbers are sampled from each tertile, excluding strata with fewer than three samples, to match group sizes and task composition. Grouping uses $K = 4$ fixed queries and remains unchanged during training. Each group starts from θ_0 and undergoes Vanilla Self-evolution with standard joint CFM, without replay or SPASE regulation, under matched training budgets and paired random seeds.

Evaluation. All groups are evaluated on the same complete task suite using the main SR protocol. To assess consistency with previously acquired knowledge, Self-Perplexity is also evaluated on 500 human demonstration pairs sampled by source task from the initialization data and excluded from diagnostic training and reference-level calibration. The score follows Eq. (1), averaged over reference pairs with 32 independent queries per modality, fixed across checkpoints. Figure 6(b) reports its percentage change relative to θ_0, computed separately for each modality and then averaged; positive values indicate greater divergence, as assessed by Self-Perplexity, between the fixed reference data and the current model's parametric knowledge.

Results. Figure 6 shows deeper early performance drops and larger increases in reference Self-Perplexity for higher-score data, despite all groups passing dual verification. The paired trends are consistent with greater interference with previously acquired knowledge and support Self-Perplexity as an indicator of adaptation instability. The high-score group nevertheless recovers during later training, indicating that these samples can provide useful learning signals despite their larger initial adaptation cost. This motivates regulating their updates rather than discarding them, complementing the controls in Appendix D. The comparison establishes an association under matched sampling conditions, rather than showing that score differences alone determine parameter displacement or forgetting.

D Effectiveness of Self-Perplexity-Guided Weighting

Experimental setup. To determine whether weighting benefits arise from sample-specific assignment rather than uniformly reducing the adaptive loss, four strategies are compared using Cosmos3-Nano on RoboCasa365. Training follows Section 4.1 and Appendix K for five rounds of 500 steps each. For each seed, a separate Vanilla Self-evolution reference run supplies a fixed sequence of verified datasets $\mathcal{D}_1, \ldots, \mathcal{D}_5$. All strategies start from the same θ_0 and receive the same data sequence, using uniform sampling from $\mathcal{T}_r = \mathcal{D}_r \cup \mathcal{D}_{r-1}$ and identical demonstration replay $\mathcal{D}_0$ in the first round. Sample orders and training-query seeds are paired across strategies, and future-round data are withheld until their corresponding round. Only the adaptive weight assignment changes; the learning-state rules and deterioration penalty remain identical, with states, reference levels, and historical baselines

Table 6: Success rates (%) after five rounds on RoboCasa365 using a shared data sequence.

Method	Atomic	Composite-Seen	Composite-Unseen	Avg.
Unweighted adaptation	77.6	37.3	14.8	44.6
Uniform weighting	80.5	40.6	16.1	47.1
Shuffled weighting	78.7	39.0	15.5	45.8
SPASE	**83.2**	**43.7**	**18.3**	**49.8**

maintained by each evolving model. This shared-data setting separates optimization effects from differences in subsequent data collection.

Weighting controls. After each state refresh, let $\mathcal{U}^m_{r,\hat{t}} = \{i \in \mathcal{T}_r : z^m_i = 0\}$ denote the targets under adaptive regulation. *Uniform weighting* replaces their Self-Perplexity-derived weights with the corresponding mean:

$$\widetilde{w}^m_{i,r,\hat{t}} = \frac{1}{|\mathcal{U}^m_{r,\hat{t}}|} \sum_{j \in \mathcal{U}^m_{r,\hat{t}}} w^m_{j,r,\hat{t}}, \qquad i \in \mathcal{U}^m_{r,\hat{t}}. \tag{13}$$

Shuffled weighting randomly permutes those weights within each modality's adaptive set, preserving their distribution while breaking the correspondence between weights and samples. Both controls refresh weights every H steps and cache them between refreshes; empty adaptive sets are skipped. *Unweighted adaptation* sets $w = 1$ as a reference, while *SPASE* retains the original sample-specific assignment. All strategies retain the deterioration penalty for protected targets. Uniform and shuffled weighting preserve the mean of the weights computed within their own run, rather than matching actual optimizer displacements across runs.

Evaluation. After the fifth round, SR is evaluated on Atomic, Composite-Seen, and Composite-Unseen tasks using the three-seed, 50-episode-per-task-per-seed protocol in Section 4.1. The overall average is computed over tasks using the same aggregation as Table 1. The comparison therefore measures task performance after an identical training budget.

Results. Table 6 shows that SPASE achieves the highest final SR across all three task splits. Uniform weighting improves upon unweighted adaptation but remains below SPASE, suggesting that uniformly reducing the adaptive loss does not fully account for the performance gains. Shuffled weighting also underperforms SPASE, supporting the usefulness of sample-specific assignment when weights are recomputed within each evolving model. Together, these comparisons support assigning learning weights according to samples' divergence from the model's parametric knowledge to improve task performance after multi-round training. Unlike the online evaluations in Tables 1 and 4, this experiment fixes the data sequence, so its final scores need not coincide with those results.

E Reliability and Data Yield of Dual Verification

Experimental setup. To assess verification reliability and the value of valid prefixes, 300 Cosmos3-Nano trajectories are sampled across the five RoboCasa365 self-evolution rounds, with 100 per task split. From these trajectories, 600 unfiltered action–video pairs are sampled independently of verifier decisions, with 200 per split. Verification follows Section 3.1 and Appendix L. Two annotators independently inspect task instructions, full execution trajectories, and aligned generated videos without seeing verifier outputs; a third adjudicates disagreements. Action eligibility requires membership in the valid prefix before the first failure, and video validity requires object-state and interaction consistency. Task requirements are checked independently of the Decomposer's output, and annotations are used only for evaluation.

Metrics. Precision is the fraction of accepted pairs satisfying the reference criterion, and recall is the fraction of valid pairs accepted within each stage's input set. Action verification is evaluated on all 600 pairs; video verification is evaluated only on action-accepted pairs, after dense-frame re-evaluation, with unresolved cases rejected. Dual-verification metrics use the original cohort and require both action and video validity. These pooled audit statistics are distinct from benchmark SR.

Table 7: Verification reliability. Video metrics are conditional on passing action verification.

Stage	Input pairs	Precision (%)	Recall (%)
Action verification	600	95.2	88.9
Video verification	420	96.2	90.0
Dual verification	600	95.0	83.9

Table 8: Data retention on the 600-pair audit cohort. Valid pairs satisfy both verification criteria.

Selection	Retained	Retention (%)	Valid pairs	Joint precision (%)
Unfiltered	600	100.0	360	60.0
Action verification only	420	70.0	330	78.6
Successful trajectories only	210	35.0	201	95.7
Dual verification	318	53.0	302	95.0

Data retention. Table 8 reports retained and jointly valid pairs from the same cohort; retention uses 600 as its denominator, while joint precision uses the retained count. The successful-trajectory-only control applies identical dual verification but additionally excludes pairs from trajectories without final simulator-confirmed success. It therefore isolates the contribution of valid prefixes without changing the video criterion.

Results. Tables 7 and 8 show that sequential verification increases the joint reliability of retained data, with reduced recall reflecting conservative selection. Compared with retaining only fully successful trajectories, valid-prefix retention provides additional supervision at comparable precision, contributing 108 additional pairs, of which 101 satisfy both validity criteria. This supports the complementary roles of filtering unreliable data and recovering useful learning signals from failed trajectories, while Table 4 assesses their downstream training benefit.

F Case Study

Figure 7 compares SPASE with Vanilla Self-evolution, illustrating more robust execution and recovery from failure in two tasks. In the moka-pot task, SPASE retries after an unsuccessful grasp and successfully picks up the pot on the next attempt. In the microwave task, SPASE avoids getting its gripper trapped by the microwave door, whereas Vanilla Self-evolution becomes stuck during the interaction.

G Efficiency

Training efficiency is evaluated on Cosmos3-Nano over one self-evolution round of 500 optimization steps. Table 9 shows that the default setting ($K = 4$) adds approximately 1.5 hours per round over Vanilla Self-evolution, an overhead of about 19%. The additional overhead mainly comes from computing Self-Perplexity over K fixed queries for action and video targets. This additional cost accompanies improved training stability (Fig. 4) and higher final task performance (Table 1). Reducing K to 2 lowers the overhead to 10% but yields weaker performance, whereas increasing K to 8 raises the overhead to 36% without further gains (Fig. 5). These results support $K = 4$ as a practical balance between training cost and performance.

H Real-World Evaluation

Setup. Cosmos3-Nano-Policy-DROID (Agarwal et al., 2026) undergoes SPASE self-evolution in RoboLab following Section 4.1. The resulting model is evaluated on a Franka arm for *Put the box into the bowl*, without additional real-world task fine-tuning. The base model, Self-evolution+ER, and SPASE are each evaluated over 50 trials.

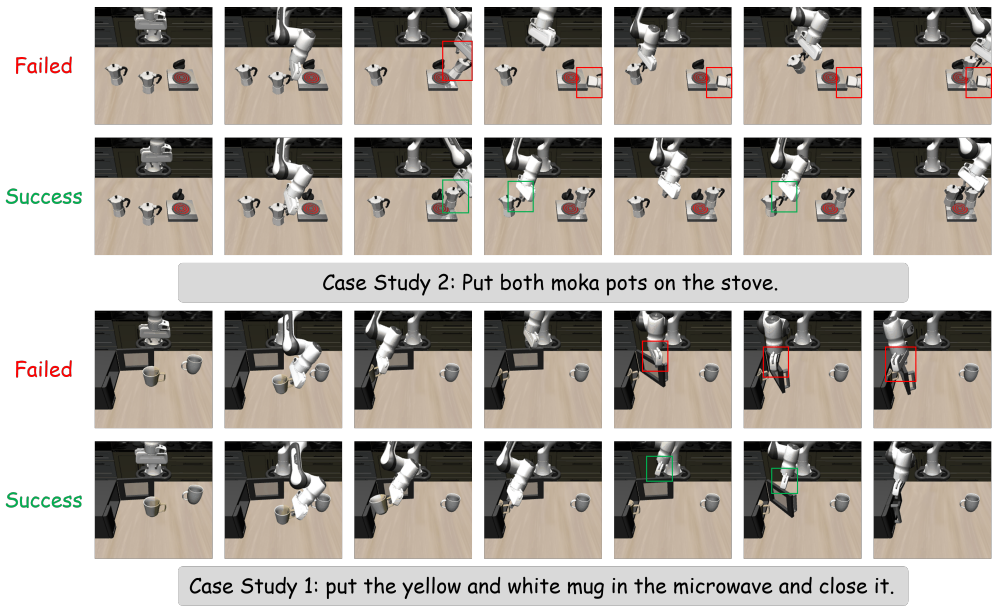


Figure 7: Case studies comparing SPASE with Vanilla Self-evolution.

Table 9: Efficiency analysis for one self-evolution round with Cosmos3-Nano.

Method	K	**Time (h)**	**Overhead**
Vanilla Self-evolution	—	8.0	—
SPASE	2	8.8	10.0%
SPASE (default)	**4**	**9.5**	**18.8%**
SPASE	8	10.9	36.3%

Results. Table 10 shows that SPASE achieves the highest observed success rate, outperforming the base model and Self-evolution+ER on this real-world task. These results suggest that the benefits of SPASE self-evolution in RoboLab can transfer to physical execution.

Qualitative result. Figure 8 presents six chronological frames from a successful real-world execution. The arm approaches and grasps the box, transports it toward the bowl, and releases it inside the bowl. This example demonstrates successful physical task execution by the SPASE-trained WAM, complementing the quantitative simulation results.

I Self-Perplexity Dynamics and Local Deterioration

This section analyzes the divergence from parametric knowledge in Secs. 2.1–2.2 in the operational, model-relative sense specified in Appendix B. The analysis first identifies the clean-target predictor encoded by the learned conditional velocity field and relates Self-Perplexity to its discrepancy from a given target. It then establishes a conditional calibration result and characterizes how adaptation can reduce this discrepancy on self-generated data while increasing it on previously learned data.

Predictive representation of parametric knowledge. For a fixed target i, let Z_i collect all conditioning, flow times, and noised inputs supplied to the WAM, and let μ_i be their distribution induced by the scoring queries. For modality m, write its flow time as $T^m \in (0,1)$ and its noised input as $Y_i^m = (1-T^m)x_i^m + T^m\epsilon^m$. The uniform flow-time distribution in Eq. (1) satisfies this open-interval condition almost surely. Define the target's conditional path velocity and the model's implied clean-target prediction by

$$u_i^m(Z_i) = \frac{Y_i^m - x_i^m}{T^m}, \qquad \widehat{x}_\theta^m(Z_i) = Y_i^m - T^m v_\theta^m(Z_i). \tag{14}$$

Table 10: Franka task success over 50 trials per method. All methods use Cosmos3-Nano-Policy-DROID as the base model.

Method	Successes / Trials	SR (%)
Base Model	41/50	82.0
Self-evolution+ER	44/50	88.0
SPASE	**46/50**	**92.0**

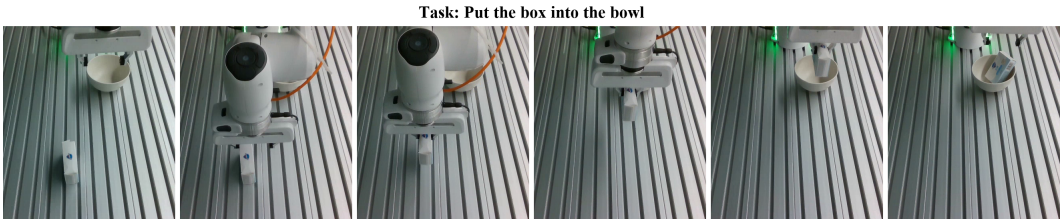


Figure 8: Franka execution after SPASE self-evolution in RoboLab, initialized from Cosmos3-Nano-Policy-DROID. The task is *Put the box into the bowl*; frames progress from left to right.

The latter is a prediction computed from the learned velocity field, rather than the endpoint obtained by integrating a sampling trajectory. All comparisons use the same evaluated coordinates and loss normalization.

Lemma 1 (Self-Perplexity as a predictive discrepancy)**.** *For every fixed target and query with* $T^m > 0$*,*

$$\ell_i^m(\theta;\xi) = \|v_\theta^m(Z_i) - u_i^m(Z_i)\|_2^2 = \frac{\|\widehat{x}_\theta^m(Z_i) - x_i^m\|_2^2}{(T^m)^2}. \tag{15}$$

Consequently, the measured finite-query Self-Perplexity satisfies the exact identity

$$\bar{\ell}_i^m(\theta) = \frac{1}{K}\sum_{k=1}^{K} \frac{\|\widehat{x}_\theta^m(Z_{i,k}) - x_i^m\|_2^2}{(T_k^m)^2}. \tag{16}$$

Its query expectation, when finite, is

$$L_i^m(\theta) := \mathbb{E}_\xi \ell_i^m(\theta;\xi) = \|v_\theta^m - u_i^m\|_{L^2(\mu_i)}^2. \tag{17}$$

For any fixed target law Q with square-integrable velocity supervision $U^m = \epsilon^m - X^m$, the unrestricted population CFM minimizer is $u_Q^m(Z) = \mathbb{E}_Q[U^m \mid Z]$. Its implied clean-target predictor satisfies

$$\widehat{x}_Q^{m,*}(Z) := Y^m - T^m u_Q^m(Z) = \mathbb{E}_Q[X^m \mid Z]. \tag{18}$$

Proof. The interpolation identity gives $u_i^m(Z_i) = \epsilon^m - x_i^m$ and $\widehat{x}_\theta^m(Z_i) - x_i^m = T^m(u_i^m(Z_i) - v_\theta^m(Z_i))$. Taking squared norms proves Eq. (15); averaging over the fixed queries or their sampling law proves Eqs. (16)–(17). Under Q, conditional expectation minimizes the squared prediction error for U^m, so the unrestricted optimum is u_Q^m. Since Y^m and T^m are included in Z and $X^m = Y^m - T^m U^m$, taking conditional expectations proves Eq. (18). □

Lemma 1 grounds the parametric-knowledge interpretation of Sec. 2.1 in the model's learned clean-target predictions: Self-Perplexity measures their time-normalized squared discrepancy from each target. The score identities apply to actual checkpoints and fixed finite-query sets; the conditional-mean characterization describes the unrestricted training objective's optimum. Using the full input Z_i accommodates joint action–video predictions with modality-specific flow times.

Calibration to a reference knowledge field. To relate this model-relative measurement to a reference predictive structure, fix a law Q and a measurable version of its conditional field u_Q^m on the evaluation inputs, independently of the evaluated checkpoints and scores. Any extension outside the support of

the reference input law must also be fixed in advance. Use the same conditioning, query construction, and normalization for the model and reference predictions. Define

$$\begin{aligned}\mathfrak{D}_i^m(Q) &:= \mathbb{E}_\xi \frac{\|\widehat{x}_Q^{m,*}(Z_i) - x_i^m\|_2^2}{(T^m)^2} = \|u_Q^m - u_i^m\|_{L^2(\mu_i)}^2, \\ \varepsilon_{i,Q}^m(\theta) &:= \|v_\theta^m - u_Q^m\|_{L^2(\mu_i)}.\end{aligned} \tag{19}$$

Here $\mathfrak{D}_i^m(Q)$ compares the target with the reference conditional predictor, independently of the evaluated checkpoint.

Corollary 1 (Calibration of target-level Self-Perplexity)**.** *If the quantities in Eq. (19) are finite, then*

$$\left|\sqrt{L_i^m(\theta)} - \sqrt{\mathfrak{D}_i^m(Q)}\right| \le \varepsilon_{i,Q}^m(\theta). \tag{20}$$

Equivalently,

$$\left[\sqrt{L_i^m(\theta)} - \varepsilon_{i,Q}^m(\theta)\right]_+^2 \le \mathfrak{D}_i^m(Q) \le \left[\sqrt{L_i^m(\theta)} + \varepsilon_{i,Q}^m(\theta)\right]^2. \tag{21}$$

The same statements hold for the fixed-query score when both reference quantities are computed using the empirical distribution of those same queries.

Proof. Apply the reverse triangle inequality in $L^2(\mu_i)$ to $v_\theta^m - u_i^m$ and $u_Q^m - u_i^m$, and use Eq. (17). The two-sided bounds follow from nonnegativity of the norms; replacing μ_i by the empirical query distribution proves the finite-query statement. □

Corollary 1 connects Self-Perplexity to a reference predictive discrepancy through the field-approximation error on μ_i. For two targets in the same modality, a difference between their square-root scores exceeding the sum of their calibration errors preserves the ordering of their reference discrepancies. Training on Q alone does not establish a small calibration error on shifted inputs. This result concerns target-to-predictor discrepancy, rather than a statistical divergence between two target distributions.

Self-Perplexity and flow matching. Fix a target distribution P, independent of the parameters being optimized, and the parameter-independent query distribution in Eq. (1). With L_i^m as in Lemma 1, its average $S_P^m(\theta) = \mathbb{E}_{i\sim P} L_i^m(\theta)$ equals the population CFM risk $R_{\mathrm{CFM},P}^m(\theta)$. The connection to marginal flow matching follows from the equivalence of the two objectives up to a parameter-independent constant (Lipman et al., 2022, Theorem 2). For modality m, let Z collect the conditioning, flow times, and noised inputs supplied to the model, and let $U^m = \epsilon^m - X^m$ denote the velocity target for the linear interpolation path. Define $u_P^m(Z) = \mathbb{E}[U^m \mid Z]$. Assuming U^m and $v_\theta^m(Z)$ are square integrable, conditional expectation gives

$$\begin{aligned}R_{\mathrm{CFM},P}^m(\theta) &:= \mathbb{E}\|v_\theta^m(Z) - U^m\|_2^2 \\ &= \underbrace{\mathbb{E}\|v_\theta^m(Z) - u_P^m(Z)\|_2^2}_{R_{\mathrm{FM},P}^m(\theta)} + \underbrace{\mathbb{E}\|U^m - u_P^m(Z)\|_2^2}_{c_P^m}.\end{aligned} \tag{22}$$

Indeed, expanding the square leaves a cross term whose conditional expectation vanishes because $\mathbb{E}[U^m - u_P^m(Z) \mid Z] = 0$. Since c_P^m is independent of θ, the CFM and FM parameter gradients agree whenever differentiation can be interchanged with expectation. The decomposition separates the mean target-level predictive discrepancy in Lemma 1 into a marginal-field error and a parameter-independent conditional-variance contribution. Since $U^m = (Y^m - X^m)/T^m$, the latter is the time-normalized conditional variance of the clean target, capturing ambiguity among targets compatible with the same model input. This aggregate decomposition does not alter the exact target-level identity in Eq. (16).

Comparing model checkpoints on fixed data. For two checkpoints θ and θ', keep the target distribution P, query distribution, and loss normalization fixed, and assume that both model velocity fields are square integrable under this law. Applying Eq. (22) at both checkpoints and subtracting gives

$$\begin{aligned}S_P^m(\theta') - S_P^m(\theta) &= R_{\mathrm{FM},P}^m(\theta') - R_{\mathrm{FM},P}^m(\theta) \\ &= \mathbb{E}\left[\|v_{\theta'}^m(Z) - u_P^m(Z)\|_2^2 - \|v_\theta^m(Z) - u_P^m(Z)\|_2^2\right].\end{aligned} \tag{23}$$

Here the parameter-independent conditional-variance term c_P^m cancels exactly, so changes in mean population Self-Perplexity quantify changes in the discrepancy between the learned velocity field and the data-induced marginal field. This identity supports monitoring changes in the consistency between fixed data and the model's parametric knowledge, including deterioration on previously learned targets as discussed in Sec. 2.1. Taking P to be the empirical distribution of a fixed sample set gives the same identity with the query expectation retained. Finite-query comparisons additionally depend on the score-estimation errors at both checkpoints, as reflected in Eq. (45). The population counterpart of the source interaction in Appendix B takes a checkpoint difference separately on each fixed source, so the two source-specific variance terms cancel separately, even when they differ across sources. Its measured finite-query value remains subject to the corresponding estimation errors.

Across different target distributions, both the marginal-field error and the conditional-variance term c_P^m can change. Absolute score differences across data sources therefore need not order their marginal-field errors and do not, by themselves, quantify a statistical divergence between probability distributions. This scope is consistent with the model-relative interpretation of divergence from parametric knowledge in Sec. 2.1; Eq. (23) instead isolates changes in marginal-field discrepancy when the target distribution is held fixed.

Self-Perplexity on reference and self-generated data. Let $R(\theta) = \sum_m S^m_{P_{\text{ref}}}(\theta)$ denote the mean population Self-Perplexity, summed over modalities, on a fixed reference distribution such as human demonstrations. Similarly, let $F(\theta) = F^{\text{a}}(\theta) + F^{\text{v}}(\theta)$, where $F^m(\theta) = S^m_{P_{\text{gen}}}(\theta)$, denote the corresponding score on a fixed distribution of self-generated data. The following argument also applies to finite averages of the measured Self-Perplexity over fixed targets and queries under the stated smoothness assumptions, without identifying those averages with population quantities. The analysis concerns the model-relative discrepancy measured by these scores; neither the population identity nor the finite-query score equates this discrepancy with task success.

Theorem 1 (Local changes in Self-Perplexity)**.** *Suppose R has an L_R-Lipschitz gradient along the segment joining θ and $\theta + d$, and write $g_R = \nabla R(\theta)$. Then*

$$|R(\theta + d) - R(\theta) - \langle g_R, d\rangle| \le \frac{L_R}{2}\|d\|_2^2. \tag{24}$$

Consequently, $\langle g_R, d\rangle > (L_R/2)\|d\|_2^2$ is sufficient for an increase in reference Self-Perplexity. For a gradient step $d = -\alpha g_F$, where $g_F = \nabla F(\theta)$ and $\alpha > 0$, this condition becomes

$$-\langle g_R, g_F\rangle > \frac{L_R\alpha}{2}\|g_F\|_2^2. \tag{25}$$

If F also has an L_F-Lipschitz gradient along this segment, then

$$F(\theta - \alpha g_F) - F(\theta) \le -\alpha\left(1 - \frac{L_F\alpha}{2}\right)\|g_F\|_2^2. \tag{26}$$

Thus, a nonzero gradient step can decrease mean Self-Perplexity on self-generated data while increasing it on reference data when Eq. (25) holds and $\alpha L_F < 2$.

Proof. The fundamental theorem of calculus gives

$$\begin{aligned} &R(\theta + d) - R(\theta) - \langle g_R, d\rangle \\ &\qquad = \int_0^1 \langle \nabla R(\theta + ud) - \nabla R(\theta), d\rangle \, du. \end{aligned} \tag{27}$$

Taking absolute values and applying the gradient Lipschitz condition bounds the integral by $\int_0^1 L_R u\|d\|_2^2 \, du = (L_R/2)\|d\|_2^2$. The lower side of this bound proves the sufficient condition for deterioration. Substituting $d = -\alpha g_F$ gives Eq. (25). Applying the upper side of the same bound to F and this displacement yields Eq. (26). □

Theorem 1 provides a local mechanism consistent with the opposing Self-Perplexity trends discussed in Sec. 2.2: the current model can become more consistent with self-generated data while becoming less consistent with previously learned data. The sufficient condition depends on gradient interaction, which is not determined by an initial score gap or distribution shift alone. Equation (24) applies to any

actual optimizer displacement satisfying the local smoothness assumption, whereas Eqs. (25)–(26) specifically describe a plain gradient step. In particular, the raw gradient need not be proportional to the displacement produced by AdamW with momentum, weight decay, and gradient clipping.

Action–video Self-Perplexity interference. For the shared model parameters, define $g_m = \nabla F^m(\theta)$ for $m \in \{\mathrm{a}, \mathrm{v}\}$ and assume that each modality risk has a locally L_m-Lipschitz gradient. Applying Eq. (24) to an illustrative joint step $d = -\alpha(c^{\mathrm{a}} g_{\mathrm{a}} + c^{\mathrm{v}} g_{\mathrm{v}})$, with fixed nonnegative coefficients $c^{\mathrm{a}}, c^{\mathrm{v}}$, gives

$$\begin{aligned} F^m(\theta + d) - F^m(\theta) &= -\alpha \sum_{n \in \{\mathrm{a},\mathrm{v}\}} c^n \langle g_m, g_n \rangle + r_m, \\ |r_m| &\le \frac{L_m}{2} \|d\|_2^2. \end{aligned} \tag{28}$$

In particular, action-only training can increase mean video Self-Perplexity when $\langle g_{\mathrm{v}}, g_{\mathrm{a}} \rangle < 0$ and the step is sufficiently small. Thus, reduced mismatch in one modality need not imply reduced mismatch in the other, motivating separate Self-Perplexity scores and learning states for action and video. Independent learning states do not themselves guarantee gradient alignment.

Stationarity and the scope of the analysis. If the reference model is initially stationary, $g_R = 0$, the first-order conflict term vanishes and smoothness alone does not determine the sign of the risk change. If R is twice continuously differentiable near θ, Taylor expansion gives

$$R(\theta + d) - R(\theta) = \frac{1}{2} d^\top \nabla^2 R(\theta) d + o(\|d\|_2^2). \tag{29}$$

Along a fixed direction of positive reference-score curvature, sufficiently small nonzero steps therefore increase reference Self-Perplexity even without a first-order conflict. These results explain how the model-relative discrepancy tracked by Self-Perplexity can deteriorate locally during adaptation, supporting the mechanism discussed alongside the V-shaped curve in Sec. 2.2. They neither imply an inevitable V-shaped success-rate curve nor establish its subsequent recovery.

J Analysis of Self-Perplexity-Aware Optimization

Building on this operational interpretation of divergence from parametric knowledge, this section analyzes how the two losses in Sec. 3.2 use Self-Perplexity to regulate targets with elevated scores and constrain subsequent deterioration on previously learned targets. The first result bounds a target's gradient contribution according to its cached Self-Perplexity, accounting for finite-query scoring and periodic weight refreshes. The second bounds increases in Self-Perplexity relative to historical baselines on retained targets and connects these bounds to the learning-state criterion and the joint training objective. The analysis assumes that the reference scales in the method are well defined and positive. In particular, computing s_r^m requires a nonempty set $\{i \in \mathcal{D}_{r-1} : z_i^m = 1\}$ at the start of the round.

J.1 Self-Perplexity-Based Control of Gradient Contributions

Fix a sample i, modality m, round r, and step t with $z_i^m = 0$, and suppress these indices where unambiguous. Write $\theta = \theta_{r,t}$, $\hat{\theta} = \theta_{r,\hat{t}}$, $s = s_r^m > 0$, $\beta = \tau^m > 0$, and $w_0 = w_{\min} \in (0, 1]$. Here β denotes the weighting temperature, not the flow time in a query. For a fresh query ξ drawn from the training query distribution, write the population-query counterpart of Self-Perplexity and its cached finite-query score as

$$L(\vartheta) = \mathbb{E}_\xi[\ell_i^m(\vartheta; \xi)], \qquad \hat{L} = \bar{\ell}_i^m(\hat{\theta}). \tag{30}$$

Let

$$w(u) = \max\left\{ w_0, \exp\left(-\frac{[u/s - 1]_+}{\beta} \right) \right\}, \qquad \hat{w} = w(\hat{L}), \qquad e = |\hat{L} - L(\theta)|. \tag{31}$$

Here $[\hat{L}/s - 1]_+$ is the cached normalized divergence gap in Eq. (3), expressing excess model-relative discrepancy above the reference Self-Perplexity level. All expectations below are conditional on the current training history, including the fixed target, current parameters, reference scale, and cached score; the fresh training query is independent of this history. Targets and noised inputs are held fixed with respect to the current parameters.

Theorem 2 (Control of adaptive gradient contributions)**.** *Write the query loss as $\ell_i^m(\theta;\xi) = \|v_\theta^m(\xi) - y^m(\xi)\|_2^2$, where $y^m(\xi)$ is the CFM velocity target and the conditioning and noised inputs are included in $v_\theta^m(\xi)$. Suppose $L(\theta) < \infty$ and the parameter Jacobian satisfies the conditional second-moment bound $\mathbb{E}_\xi \|\partial v_\theta^m(\xi)/\partial\theta\|_{\mathrm{op}}^2 \le B^2$. For the stop-gradient weighted contribution $G(\xi) = \hat{w}\nabla_\theta \ell_i^m(\theta;\xi)$,*

$$\begin{aligned}\mathbb{E}_\xi\|G(\xi)\|_2 &\le 2B\hat{w}\sqrt{L(\theta)} \\ &\le 2B\sqrt{L(\theta)}\min\left\{1, w(L(\theta)) + \frac{e}{s\beta}\right\}.\end{aligned} \tag{32}$$

In the exact current-score case $e = 0$, this further implies

$$\mathbb{E}_\xi\|G(\xi)\|_2 \le 2B\max\left\{w_0\sqrt{L(\theta)}, \sqrt{s}\,C_\beta\right\}, \qquad C_\beta = \begin{cases}1, & 0 < \beta \le 2, \\ \sqrt{\beta/2}\,\exp(1/\beta - 1/2), & \beta > 2.\end{cases} \tag{33}$$

Proof. Writing $J_\theta^m(\xi) = \partial v_\theta^m(\xi)/\partial\theta$, the chain rule gives

$$\|\nabla_\theta \ell_i^m(\theta;\xi)\|_2 \le 2\|J_\theta^m(\xi)\|_{\mathrm{op}}\sqrt{\ell_i^m(\theta;\xi)}. \tag{34}$$

Since $\hat{w}$ is fixed under the conditional expectation, the Cauchy–Schwarz inequality yields

$$\begin{aligned}\mathbb{E}_\xi\|G(\xi)\|_2 &\le 2\hat{w}\sqrt{\mathbb{E}_\xi\|J_\theta^m(\xi)\|_{\mathrm{op}}^2}\sqrt{\mathbb{E}_\xi \ell_i^m(\theta;\xi)} \\ &\le 2B\hat{w}\sqrt{L(\theta)}.\end{aligned} \tag{35}$$

This step does not require the Jacobian and the residual to be independent. The function w is continuous and piecewise differentiable, with derivative magnitude at most $1/(s\beta)$ wherever differentiable. It is therefore $1/(s\beta)$-Lipschitz on $[0,\infty)$. Together with $w \le 1$, this gives $\hat{w} \le \min\{1, w(L(\theta)) + e/(s\beta)\}$ and proves Eq. (32). For $e = 0$, let $q = L(\theta)/s$ and consider $f(q) = \sqrt{q}\exp(-[q-1]_+/\beta)$. On $[0,1]$, its maximum is 1. For $q > 1$, its logarithmic derivative is $1/(2q) - 1/\beta$. Thus, its maximum is attained at $q = 1$ when $\beta \le 2$ and at $q = \beta/2$ when $\beta > 2$, with value C_β. Using $w(L)\sqrt{L} = \max\{w_0\sqrt{L}, \sqrt{s}\,f(L/s)\}$ establishes Eq. (33). □

The Jacobian and loss must use the same normalization. For a mean squared error over D valid coordinates, the theorem applies after dividing both the prediction and target vectors by $\sqrt{D}$ and defining B for the normalized prediction. Together with Lemma 1, the bound links a target's discrepancy from the model's encoded clean-target predictions to its expected gradient contribution under the stated Jacobian condition. A large score alone does not imply a large gradient, since the Jacobian also matters. In the exact-score case, the exponential branch has a bounded envelope, whereas the positive weight floor leaves a term proportional to $w_0\sqrt{L(\theta)}$. The weighting attenuates each target's gradient contribution according to its cached score; its relation to the current population loss is controlled by e in Eq. (32). The positive floor does not provide a globally bounded contribution as the loss tends to infinity. The result also does not imply that the summed gradient norm or the actual AdamW parameter displacement must decrease, because weighting changes gradient cancellation and the optimizer transforms the resulting gradient.

Connection to reference Self-Perplexity. For the differentiable reference score R in Appendix I, consider an illustrative single-target gradient displacement $d(\xi) = -\alpha G(\xi)$ with $\alpha > 0$. Cauchy–Schwarz and Theorem 2 give

$$\mathbb{E}_\xi\,|\langle\nabla R(\theta), d(\xi)\rangle| \le 2\alpha B\hat{w}\sqrt{L(\theta)}\,\|\nabla R(\theta)\|_2. \tag{36}$$

Thus, the Self-Perplexity-dependent weight also controls a bound on this target's contribution to the first-order change in reference mismatch. This statement concerns the linear term in Eq. (24); the curvature remainder and the full expected score change are not bounded by Eq. (36).

Finite queries and delayed refreshes. The discrepancy e in Eq. (32) separates into score-estimation error and model drift. If L is G_L-Lipschitz on the segment from $\hat{\theta}$ to θ, the triangle inequality gives

$$\begin{aligned}e &\le \underbrace{|\hat{L} - L(\hat{\theta})|}_{\varepsilon_{\mathcal{Q}}} + G_L\|\theta - \hat{\theta}\|_2 \\ &\le \varepsilon_{\mathcal{Q}} + G_L\sum_{j=\hat{t}}^{t-1}\|\theta_{r,j+1} - \theta_{r,j}\|_2.\end{aligned} \tag{37}$$

This bound uses the actual parameter displacements and contains at most $H-1$ updates between refreshes. It is conditional on the realized query-estimation error $\varepsilon_{\mathcal{Q}}$ and does not assume that this error is small for a particular value of K. Because the fixed queries are reused in training decisions, a probabilistic guarantee for $\varepsilon_{\mathcal{Q}}$ would require additional assumptions controlling this dependence.

J.2 Controlling Self-Perplexity Deterioration on Retained Targets

For a previously learned target, an increase in Self-Perplexity relative to its historical baseline indicates increased divergence from the current model's parametric knowledge in the operational sense of Sec. 2.1, evaluated under the same queries. By Lemma 1, this increase is exactly an increase in the time-normalized discrepancy between the fixed target and the model's clean-target predictions on those queries. The deterioration penalty in Eq. (8) controls this increase through the constituent query-level errors. The following result bounds both the target's Self-Perplexity increase and the fraction of queries exhibiting larger deterioration.

Theorem 3 (Control of Self-Perplexity deterioration)**.** *Consider a target (i,m) with $i \in \mathcal{T}_r$ and $z_i^m = 1$, evaluated using its K fixed queries. Write $\Delta_k(\theta) = \ell_i^m(\theta;\xi_{i,k}^m) - b_i^m(\xi_{i,k}^m)$, $\bar{b}_i^m = K^{-1}\sum_{k=1}^K b_i^m(\xi_{i,k}^m)$, and $a^m = \delta^m\kappa^m$, where $\kappa^m > 0$ and $\delta^m \geq 0$. Let $h_i^m(\theta)$ denote Eq. (8) evaluated at parameters θ. For every θ,*

$$\left[\bar{\ell}_i^m(\theta) - \bar{b}_i^m - a^m\right]_+ \leq \sqrt{2\kappa^m h_i^m(\theta)}. \tag{38}$$

Moreover, for every $u > 0$,

$$\frac{1}{K}\#\left\{k : \Delta_k(\theta) > a^m + u\right\} \leq \frac{2\kappa^m h_i^m(\theta)}{u^2}. \tag{39}$$

Proof. The positive-part function is convex, and the Cauchy–Schwarz inequality gives

$$\begin{aligned}\left[\frac{1}{K}\sum_{k=1}^K \Delta_k - a^m\right]_+ &\leq \frac{1}{K}\sum_{k=1}^K [\Delta_k - a^m]_+ \\ &\leq \sqrt{\frac{1}{K}\sum_{k=1}^K [\Delta_k - a^m]_+^2} = \sqrt{2\kappa^m h_i^m(\theta)}.\end{aligned} \tag{40}$$

This establishes Eq. (38). For each query, $\mathbf{1}\{\Delta_k > a^m + u\} \leq [\Delta_k - a^m]_+^2/u^2$. Averaging this inequality establishes Eq. (39). □

Thus, if the attained penalty satisfies $h_i^m(\theta) \leq \varepsilon$, then

$$\bar{\ell}_i^m(\theta) \leq \bar{b}_i^m + \delta^m\kappa^m + \sqrt{2\kappa^m\varepsilon}. \tag{41}$$

In particular, $h_i^m(\theta) = 0$ implies that every evaluated query remains within its individual historical baseline plus the margin. The result holds for every parameter choice and fixed query realization, including parameters obtained by adaptively optimizing these same queries.

Connection to the learning-state criterion. Suppose a self-generated target (i,m) first enters the protected state in round r_b by satisfying the reference Self-Perplexity criterion in Sec. 3.2. Its recorded baseline then satisfies $\bar{b}_i^m \leq s_{r_b}^m$. While the target remains in replay and protected, substituting this inequality into Eq. (38) yields

$$\bar{\ell}_i^m(\theta) \leq s_{r_b}^m + \delta^m\kappa^m + \sqrt{2\kappa^m h_i^m(\theta)}. \tag{42}$$

This connects the two uses of Self-Perplexity: reaching the reference level establishes the recorded fit, and a small subsequent penalty limits how far the score can exceed that level. The bound uses the reference level at entry into the protected state, which need not equal the reference level in a later round. Human demonstrations initialized with $z_i^m = 1$ need not satisfy this entry criterion; their guarantee remains relative to their recorded baseline as in Eq. (41).

The query-level construction prevents improvements on some queries from canceling deterioration on others before thresholding. For example, when $K = 2$ and the margin is zero, changes $(c, -c)$

with $c > 0$ leave average Self-Perplexity unchanged but incur penalty $c^2/(4\kappa^m)$. The fixed historical baseline also keeps the bound anchored to the same recorded fit across rounds in which the target remains protected, so the tolerated margin does not accumulate merely because additional rounds are performed. Separate action and video penalties provide this bound for each protected modality individually.

Connection to the joint objective. Fix a round and training step, including the learning states, cached weights, and historical baselines. Let p denote the sample distribution used by the mini-batch sampler, and define $q_m = \Pr_{i\sim p}(z_i^m = 1)$ and $H_m(\theta) = \mathbb{E}_{i\sim p}[h_i^m(\theta) \mid z_i^m = 1]$ for modalities with $q_m > 0$. Taking the full sampling expectation of Eq. (10) gives

$$J(\theta) = A(\theta) + \lambda_{\text{stable}} \sum_{m:q_m>0} q_m H_m(\theta), \qquad A(\theta) \geq 0, \tag{43}$$

where $A(\theta)$ is the expected adaptive loss, including its fresh-query expectation. For finite $J(\theta)$ and $\lambda_{\text{stable}} > 0$, nonnegativity implies $H_m(\theta) \leq J(\theta)/(\lambda_{\text{stable}} q_m)$. Taking expectations in Eq. (38) and applying Jensen's inequality therefore gives

$$\mathbb{E}_{i\sim p}\Big[\big[\bar{\ell}_i^m(\theta) - \bar{b}_i^m - \delta^m\kappa^m\big]_+ \,\Big|\, z_i^m = 1\Big] \leq \sqrt{\frac{2\kappa^m J(\theta)}{\lambda_{\text{stable}} q_m}}. \tag{44}$$

This bound connects average excess deterioration to the attained expected objective and the sampling probability of protected targets. It concerns the full sampling expectation, not an arbitrary observed mini-batch loss. Since the attained parameters and objective also depend on λ_{stable}, it does not imply that increasing this coefficient necessarily improves retention in an actual training run.

Population scope. For a target (i, m), let $\theta_{b,i}^m$ be the parameters at which its baseline was recorded, so that $\bar{b}_i^m = \bar{\ell}_i^m(\theta_{b,i}^m)$. Define $L_i^m(\vartheta) = \mathbb{E}_\xi \ell_i^m(\vartheta;\xi)$ and $e_i^m(\vartheta) = |L_i^m(\vartheta) - \bar{\ell}_i^m(\vartheta)|$. Assume the population query losses at θ and $\theta_{b,i}^m$ are finite. Adding and subtracting the two fixed-query scores and applying Eq. (38) yields the deterministic extension

$$\begin{aligned} L_i^m(\theta) - L_i^m(\theta_{b,i}^m) \leq \delta^m\kappa^m + \sqrt{2\kappa^m h_i^m(\theta)} \\ + e_i^m(\theta) + e_i^m(\theta_{b,i}^m). \end{aligned} \tag{45}$$

Thus, extending finite-query control to the population query loss requires control of both score-estimation errors. This identity-based bound remains valid for adaptively selected parameters, but the reused queries do not themselves guarantee small estimation errors. Historical baselines may also correspond to different parameter snapshots for different targets, so they need not define a single reference model in Eq. (22).

The guarantees above constrain the growth of model-relative discrepancy measured by Self-Perplexity on retained targets under their stated conditions. The penalty is a soft constraint, so its presence in the objective does not itself ensure that its value stays small or decreases at every update. With $\mathcal{T}_r = \mathcal{D}_r \cup \mathcal{D}_{r-1}$, the training objective does not control the penalty on earlier targets after they leave replay. Preserving these finite-query errors alone does not establish monotonic task success or the absence of behavioral forgetting.

K Experimental Details

This section provides the key configurations used in our experiments. All experiments are conducted on 8 NVIDIA A100 GPUs with 80GB memory.

K.1 Model Initialization before Self-Evolution

RoboCasa365. Cosmos3-Nano, DreamZero, and LingBot-VA are fine-tuned on the Human300 demonstration dataset for 40k optimization steps each. For each model, the resulting checkpoint serve as the initialization θ_0 for self-evolution.

LIBERO-10. Cosmos3-Nano is fine-tuned on the LIBERO-10 demonstration dataset for 1,500 optimization steps before self-evolution. The fine-tuning budgets on the same dataset are 2k steps for DreamZero and 3.5k steps for LingBot-VA.

Table 11: Optimization configuration for self-evolution.

Configuration	Value
Optimizer	FusedAdam in AdamW mode
Weight decay	0.05
Learning rate	5×10^{-5}
Per-device batch size	16
Gradient accumulation steps	16
Effective batch size	2,048
Learning-rate schedule	Linear warmup + linear decay
Warmup steps	200
Gradient clipping	Global L_2 norm, maximum 1.0
Training precision	BF16

RoboLab. Cosmos3-Nano and DreamZero are initialized from the publicly released Cosmos3-Nano-Policy-DROID and DreamZero-DROID checkpoints, respectively, which are trained on DROID (Khazatsky et al., 2024). LingBot-VA is initialized from the public LingBot-VA-Base checkpoint and fine-tuned on DROID with a budget of 50k optimization steps.

K.2 Baseline Configurations

LoRA+ER. The LoRA configuration uses an adapter rank of 16 and a scaling parameter $\alpha_{\mathrm{LoRA}} = 32$, giving an adapter scaling multiplier of 2.

L2-SP+ER. Following L2-SP (Xuhong et al., 2018), the model is regularized toward its parameters at the start of each round:

$$\mathcal{L}_{r,t}^{\text{L2-SP}} = \mathbb{E}_{i\sim\mathcal{T}_r}\left[\sum_{m\in\{\mathrm{a},\mathrm{v}\}} \ell_{i,r,t}^{m}\right] + \frac{\lambda_{\mathrm{param}}}{2}\left\|\theta_{r,t} - \theta_{r,0}\right\|_2^2, \tag{46}$$

where $\mathcal{T}_r = \mathcal{D}_r \cup \mathcal{D}_{r-1}$ and $\ell_{i,r,t}^{m}$ is the standard CFM loss defined in the main text. The penalty sums squared differences over trainable parameters, with an initial coefficient $\lambda_{\mathrm{param}} = 10^{-3}$. The reference $\theta_{r,0}$ is frozen within each round and refreshed at the next round.

Velocity Regularization+ER. Inspired by the velocity-field regularizer in ORW-CFM-W2 (Fan et al., 2025), this baseline penalizes the MSE between the current and frozen round-start models' action and video velocity predictions on replay samples:

$$\begin{aligned}\mathcal{L}_{r,t}^{\mathrm{out}} = \mathbb{E}_{i\sim\mathcal{T}_r}&\left[\sum_{m\in\{\mathrm{a},\mathrm{v}\}} \ell_{i,r,t}^{m}\right] \\ &+ \lambda_{\mathrm{out}}\mathbb{E}_{i\sim\mathcal{D}_{r-1},\,\xi_i}\left[\sum_{m\in\{\mathrm{a},\mathrm{v}\}} \mathrm{MSE}\Big(v_{\theta_{r,t}}^{m}(\cdot), \mathrm{sg}[v_{\theta_{r,0}}^{m}(\cdot)]\Big)\right],\end{aligned} \tag{47}$$

where ξ_i collects the sampled flow timesteps and noises, and $(\cdot)$ denotes identical conditioning, noised action–video inputs, and timesteps for both models. The MSE is averaged over valid prediction elements separately for each modality. The reference $\theta_{r,0}$ remains frozen within each round, and the additional forward pass is evaluated without gradients. The initial coefficient is $\lambda_{\mathrm{out}} = 0.1$ for both modalities. Only the velocity-field penalty is adapted; training retains standard CFM supervision without reward weighting.

K.3 Self-evolution Training

Optimization. Table 11 summarizes the optimization configuration for self-evolution.

Data and Rollouts. Each task is rolled out 10 times per self-evolution round. For the first round, $\mathcal{D}_0$ contains five expert trajectories uniformly sampled per task from the human demonstrations used to

initialize θ_0, excluding diagnostic evaluation trajectories. The same subset is shared across methods using demonstration replay for each random seed.

L Multimodal Large Language Model as Judge

Execution-Grounded Dual Verification filters self-generated trajectories before they are used for training (Sec. 3.1). The module comprises three roles: a *Decomposer*, an *Action Verifier*, and a *Video Verifier*. All three roles are instantiated with the same MLLM, Gemini 2.5 Pro, using separate calls with role-specific prompts. The procedure retains valid execution prefixes and then removes action–video pairs whose imagined videos are inconsistent with actual execution. In the templates below, braces denote text placeholders, while bracketed image markers denote actual image attachments in the multimodal user message. Each frame is preceded by a text label identifying its source and timestamp in seconds; execution frames use trajectory-relative timestamps, while paired video frames share a time origin at the start of the action window. For uncertain video comparisons, the calling procedure supplies denser frames from the same two clips and repeats the verification.

Decomposer. Given the task instruction and initial observation C_j, the Decomposer produces the ordered subgoals $\mathcal{S}_j$. Each subgoal specifies an observable condition that the Action Verifier can subsequently assess.

Decomposer Prompt

System Message:
You decompose robot manipulation tasks into ordered, visually verifiable subgoals. Use only the task instruction and initial observation. Apply the following rules:

1. **Ground the decomposition.** Identify task-relevant objects and relations from the instruction and image. Do not invent unseen objects, hidden properties, or objectives absent from the instruction.
2. **Use observable conditions.** Describe each subgoal as a concrete state or interaction whose completion can be checked visually, such as holding an object or placing it inside a container. Do not require an exact arm trajectory when different motions can satisfy the same condition.
3. **Preserve task order.** Arrange the essential conditions in a valid execution order and include the final condition required by the instruction. Avoid redundant subgoals and unrelated intermediate actions.

Return only a JSON object with a `subgoals` array. Each entry must contain an integer `id`, starting from 1, and a concise `condition` describing what must be visibly achieved. The array order defines the subgoal order.

User Message:
Task instruction: `{task_instruction}`
Initial observation: [attached initial-observation image]
Decompose this task into ordered, visually verifiable subgoals.

Action Verifier. Given C_j, $\mathcal{S}_j$, and the recorded execution video V_j^{exec}, the Action Verifier checks subgoal completion and locates the corresponding times. Only the longest consecutively valid prefix is retained; a failed check excludes its segment and all subsequent samples, even if execution later recovers.

Action Verifier Prompt

System Message:
You verify robot task execution from timestamped video frames. Use the supplied instruction, initial observation, and ordered subgoals as the reference. Apply the following rules:

1. **Verify sequentially.** For each subgoal, inspect all supplied frames after the previous verified completion, or from the video start for the first subgoal. Check both the completion condition and the intervening execution, rather than only the final frame.
2. **Require visible execution evidence.** Verify a subgoal only when its condition is visibly achieved without object drops, failed grasps, or unintended collisions in that segment. Gripper closure alone does not establish a successful grasp; use the visible object–gripper relation and subsequent motion as evidence. Distinguish task-required contact from unintended collisions.
3. **Stop at the first failed check.** If the condition is not achieved, an execution failure occurs, or the frames do not establish completion, mark the subgoal as unverified and stop. Do not infer success from intended actions or use later recovery to retain the failed segment or subsequent segments.

4. **Ground temporal locations.** For a verified subgoal, report the earliest supplied timestamp that establishes completion, with brief visual evidence. Do not invent intermediate timestamps or events between sampled frames.

Return only a JSON object with an `assessments` array and `valid_prefix_end`. For each assessed subgoal, report its `id`, Boolean `verified`, `completion_time`, and brief `evidence`. Use `null` for the completion time of an unverified subgoal. Set `valid_prefix_end` to the completion time of the last consecutively verified subgoal, or `null` if none is verified.

User Message:
Task instruction: `{task_instruction}`
Initial observation: [attached initial-observation image]
Ordered subgoals: `{ordered_subgoals}`
Execution frames (repeat the following block in chronological order):
Execution frame, time: `{timestamp}` s; [attached execution-frame image]
Verify the subgoals and identify the end of the longest valid execution prefix.

Video Verifier. The Video Verifier compares each retained sample's generated video V_i with its aligned execution clip V_i^{exec} for object state consistency and interaction consistency. Uncertain cases are re-evaluated with denser frames; the action–video pair is retained only if both criteria receive a final *pass*.

Video Verifier Prompt

System Message:
You compare a model-generated video with the actual execution video for the same robot action window. The two clips are temporally aligned; use the supplied timestamps to compare corresponding stages. Evaluate the two criteria independently using the following rules:

1. **Compare corresponding evidence.** Use the source labels to distinguish generated frames from execution frames and compare the same task-relevant objects at matching timestamps. Inspect the sequence rather than only the final state.
2. **Check object state consistency.** Determine whether key objects have matching states and spatial relations, such as being held, supported by a surface, or inside a container. A shared final state does not override a visible state contradiction earlier in the window.
3. **Check interaction consistency.** Determine whether grasping, contacting, releasing, and resulting object motion agree across the clips. Check the visible interaction sequence, rather than inferring agreement from similar object appearance.
4. **Calibrate the judgment to the evidence.** Ignore differences in rendering style or texture that do not affect either criterion. Use `pass` for visibly supported consistency, `fail` for a clear task-relevant contradiction, and `uncertain` when occlusion or sparse frames prevent a reliable judgment. Neither successful real execution nor a plausible generated video is sufficient for a pass.

Return only a JSON object with `object_state_consistency`, `interaction_consistency`, and `decision`. For each criterion, provide a `label`, a brief `evidence` description, and the relevant `timestamps`. Set `decision` to `pass` only if both criteria pass, to `fail` if either fails, and to `uncertain` otherwise. Report only timestamps supplied with the images. If denser frames are supplied for re-evaluation, apply the same criteria and retain `uncertain` if the additional evidence remains insufficient.

User Message:
Task instruction: `{task_instruction}`
Conditioning observation: [attached conditioning-observation image]
Aligned frame pairs (repeat the following block in chronological order):
Generated frame, time: `{timestamp}` s; [attached generated-frame image]
Execution frame, time: `{timestamp}` s; [attached execution-frame image]
Assess object state consistency and interaction consistency for this pair.

M TRAINING ALGORITHM

Computing the reference level s_r^m assumes $\{i \in \mathcal{D}_{r-1} : z_i^m = 1\} \neq \emptyset$ for every round and modality, a condition satisfied in all reported experiments.

Algorithm 1 SPASE: Multi-Round Self-Evolution of World Action Models

Require: Initial WAM θ_0; demonstrations $\mathcal{D}_0$; rollout tasks, simulator, and verifiers in Sec. 3.1; R rounds, T steps per round; optimizer and mini-batch sampler; K queries, evaluation interval H; η, $w_{\min}$, τ^m, δ^m, and λ_{stable}, with $m \in \{\text{a}, \text{v}\}$.

Ensure: Updated WAM θ_R.

Initialize $z_i^m \leftarrow 1$, $b_i^m(\xi_{i,k}^m) \leftarrow \ell_i^m(\theta_0; \xi_{i,k}^m)$ for queries $\{\xi_{i,k}^m\}_{k=1}^K$.
for $r = 1$ **to** R **do**
  $\theta_{r,0} \leftarrow \theta_{r-1}$.
  Obtain $\mathcal{D}_r$ through Execution-Grounded Dual Verification.
  $\mathcal{T}_r \leftarrow \mathcal{D}_r \cup \mathcal{D}_{r-1}$; initialize fixed queries and $z_i^m \leftarrow 0$ for new targets.
  Compute s_r^m by Eq. (2) and fix it within the round.
  If $r = 1$, set $\kappa^m \leftarrow s_1^m$.
  for $t = 0$ **to** T **do**
    if $t \bmod H = 0$ **then**
      $\hat{t} \leftarrow t$; evaluate $\bar{\ell}_i^m(\theta_{r,\hat{t}})$ for all targets in $\mathcal{T}_r$ by Eq. (1).
      for each target (i, m) with $z_i^m = 0$ **do**
        if $\bar{\ell}_i^m(\theta_{r,\hat{t}}) \leq s_r^m$ **then**
          $z_i^m \leftarrow 1$; fix $b_i^m(\xi_{i,k}^m) \leftarrow \ell_i^m(\theta_{r,\hat{t}}; \xi_{i,k}^m)$.
        else
          Refresh $d_{i,r,\hat{t}}^m$ and $w_{i,r,\hat{t}}^m$ by Eqs. (3)–(4).
        end if
      end for
    end if
    if $t < T$ **then**
      Sample a mini-batch from $\mathcal{T}_r$.
      for each sampled target (i, m) **do**
        if $z_i^m = 0$ **then**
          $\mathcal{L}_{\text{adapt},i,r,t}^m \leftarrow w_{i,r,\hat{t}}^m \ell_{i,r,t}^m$, $\mathcal{L}_{\text{penalty},i,r,t}^m \leftarrow 0$.
        else
          $\mathcal{L}_{\text{adapt},i,r,t}^m \leftarrow 0$.
          $$\mathcal{L}_{\text{penalty},i,r,t}^m \leftarrow \frac{1}{K} \sum_{k=1}^{K} \frac{\left[\ell_i^m(\theta_{r,t}; \xi_{i,k}^m) - b_i^m(\xi_{i,k}^m) - \delta^m \kappa^m\right]_+^2}{2\kappa^m}.$$
        end if
      end for
      $\mathcal{L}_{i,r,t} \leftarrow \sum_{m \in \{\text{a},\text{v}\}} \left[\mathcal{L}_{\text{adapt},i,r,t}^m + \lambda_{\text{stable}} \mathcal{L}_{\text{penalty},i,r,t}^m\right]$.
      Update $\theta_{r,t}$ to $\theta_{r,t+1}$ using the mini-batch average of $\mathcal{L}_{i,r,t}$.
    end if
  end for
  $\theta_r \leftarrow \theta_{r,T}$.
end for
return θ_R.