

TSAIL Invited Talk



Understanding Diffusion Reinforcement Learning through the Lens of Policy Optimization in Continuous Time and Space

Understanding Diffusion RL through Policy Optimization in Continuous Time and Space

Zhenglin Wan

HPC-AI Lab, National University of Singapore

1. A Unified Mental Model through the MSE Interface

Insight | Diffusion RL algorithms use MSE as their training interface.



Output force F → Per-sample parameter gradient P → Aggregate gradient A

Three quantities

1. Output force F

Force magnitude at each state.

2. Per-sample parameter gradient P

Magnitude after mapping force to parameters.

3. Aggregate gradient across states A

Magnitude after aggregating state gradients.

Two ratios linking these quantities

1. Per-sample response P/F

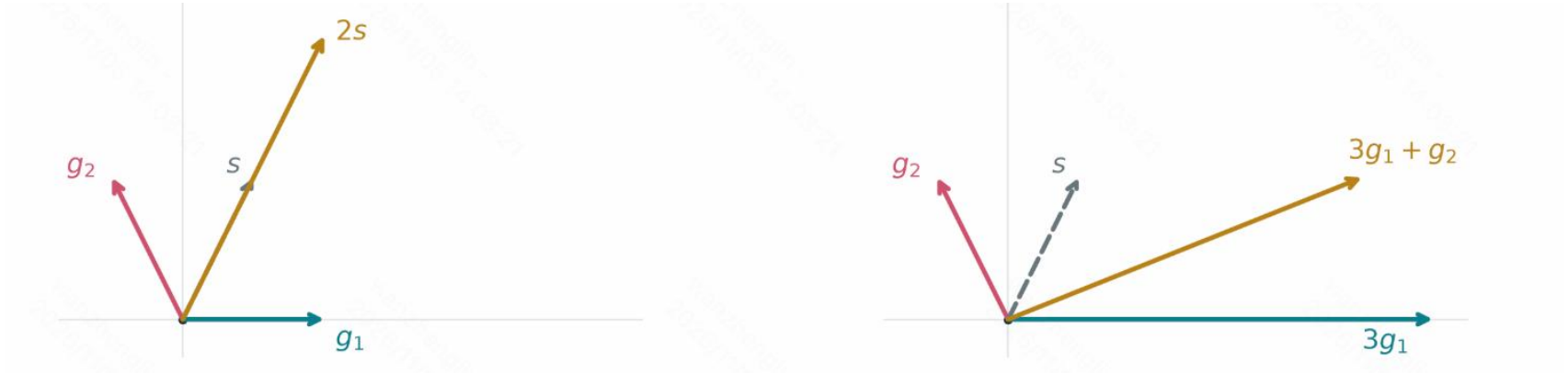
Parameter-gradient magnitude per unit force.

2. Aggregation retention A/P

Fraction of gradients retained in aggregation.

Insight 1 | Forces admit vector addition and decomposition

$$J^T(f_1 + f_2) = J^T f_1 + J^T f_2$$



Insight 2 | Relative scale matters more than absolute scale

$$m_n = \beta_1 m_{n-1} + (1-\beta_1) h_n,$$
$$q_n = \beta_2 q_{n-1} + (1-\beta_2) h_n^2,$$
$$\theta_{n+1} = (1-\eta_n \lambda_w)\theta_n - \eta_n \frac{\widehat{m}_n}{\sqrt{\hat{q}_n} + \epsilon}.$$

Modern optimizers such as AdamW remove global scaling.

Relative scales remain: they change the resultant direction.

Insight 3 | Reward-improving forces must survive aggregation

Analyze how each state's action is pulled or pushed and how forces map to parameters.
Then examine which components dominate or cancel after aggregation.
This is the common starting point for Reverse RL and Forward RL.

2. Understanding Reverse-Process RL (Flow-GRPO Family)

2.1 Preliminaries: Denoising Transitions as a Policy

ODE-to-SDE: Actions gain a density

state $s_k = (c, t_k, x_k)$ action $a_k = x_{k+1}$

$\pi_\theta(a_k \mid s_k) = N(\mu_\theta(s_k), \sigma_k^2 \delta_k I)$

Endpoint reward → Group advantage → Per-step PPO

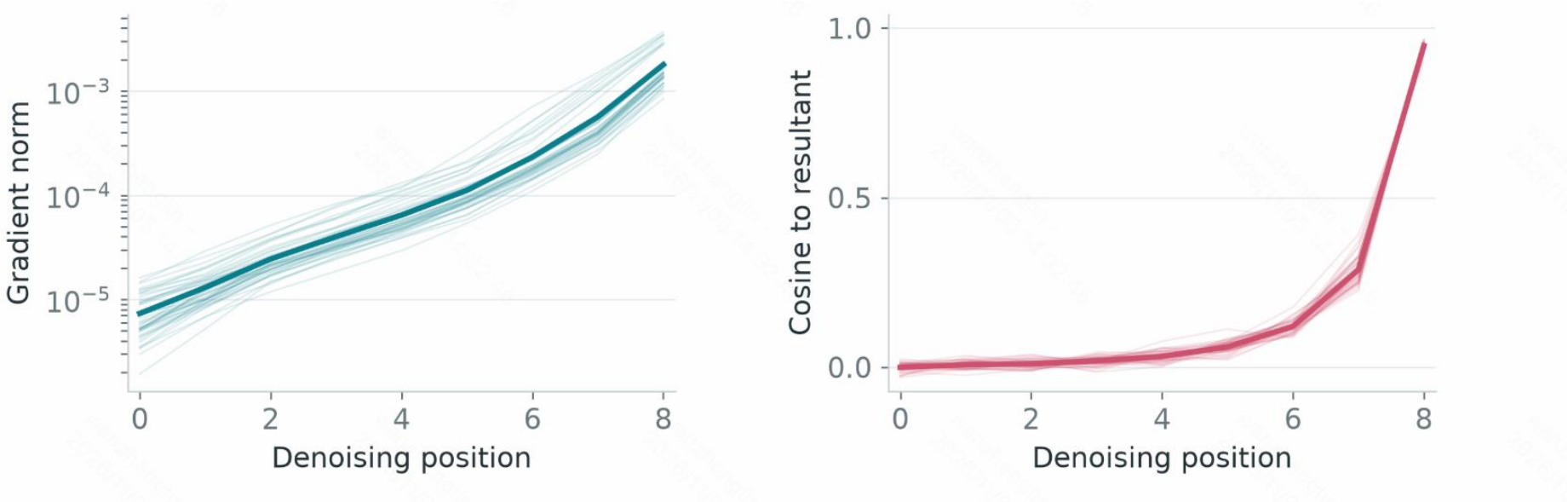
Local MSE interface:

$\ell_{i,k} = A_i\, w_k\, \|v_\theta - v_{i,k}^{tgt}\|^2$

$f_{i,k} = 2A_i w_k (v_{i,k}^{tgt} - v_\theta)$

$w_k = \delta_k b_k^2/(2\sigma_k^2) \propto (1-t_k)/t_k$

Original SDE, fixed δ, α



High noise → Low noise

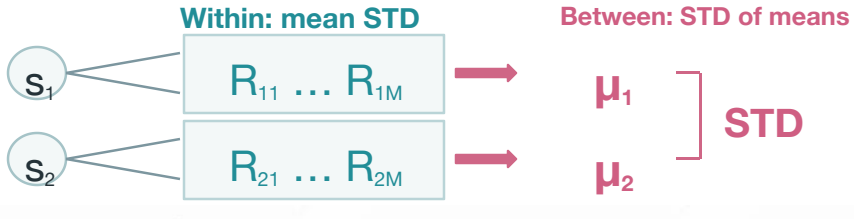
Observation | Low-noise forces have a larger share in original Flow-GRPO.

2.2 Denoising Is a Special MDP

Observation | Denoising prior: early actions affect reward more than late ones.

Early actions can still change reward substantially. Late rewards largely inherit state quality.

Same prompt: fix states, branch to rewards





What enters the group advantage?

$$A^{local} = Q(s,a) - V(s) \qquad A^{group} = R - b_G$$

$$\Delta A = A^{group} - A^{local}$$

$$= [V(s) - b_G] + [R - Q(s,a)]$$

State-baseline mismatch Monte Carlo noise

$$E[\Delta A_i \mid S,c] = V(s_i) - \bar{V}$$

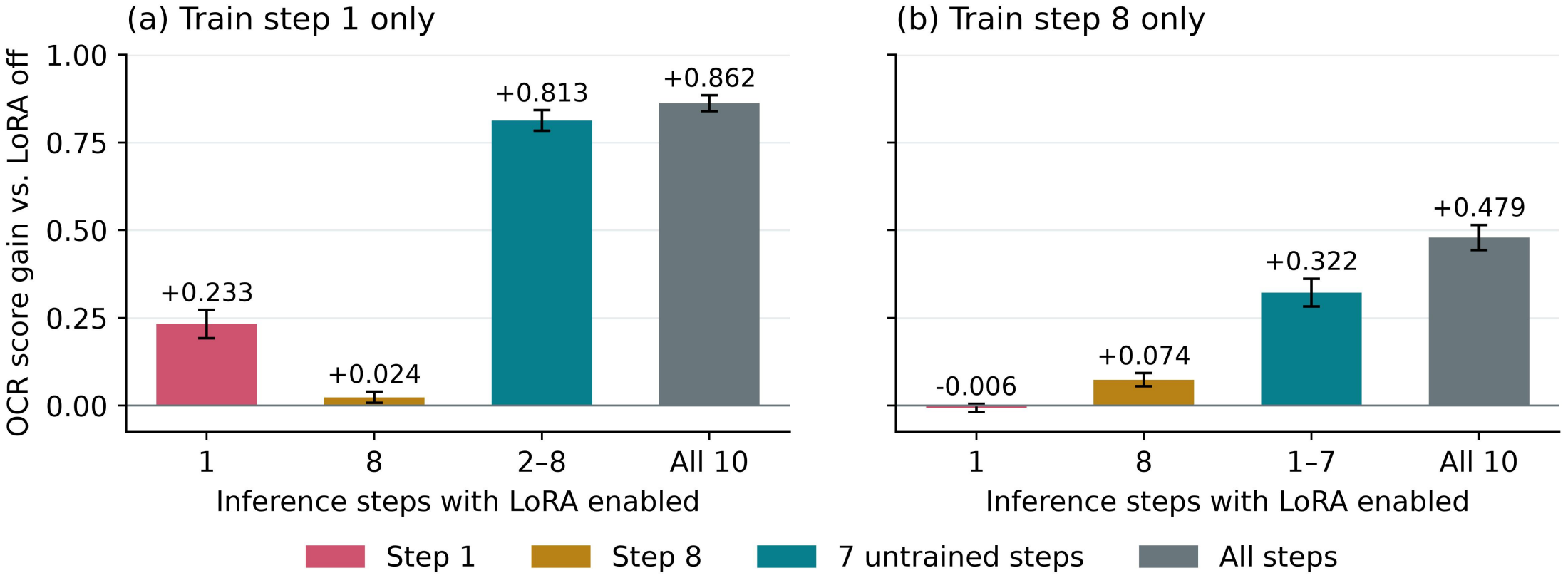
$$\text{mean}_i\{E[\Delta A_i \mid S,c]^2\} = \text{Var}_i[V(s_i)]$$

Greater between-group variance in state values means greater baseline mismatch.

Insight | Unlike LLM RL, noise levels offer structure: high-noise windows may carry more action credit.

2.3 Gains Extend beyond the Trained Timesteps

Train only step 1 or 8. During 10-step inference, toggle the same LoRA at each step to locate the gains.



Evaluate after 120 rollouts on 64 fixed prompts. Gains relative to all-off baseline 0.064. Error bars: ±1 SEM.

Insight | Timestep selection chooses gradient sources, not where gains can appear.

Shared parameters can carry useful high-noise action signals to untrained timesteps.

2.4 Understanding Flow-GRPO-Fast

High-noise window: hard weighting

Random m-step window: N → m update samples.

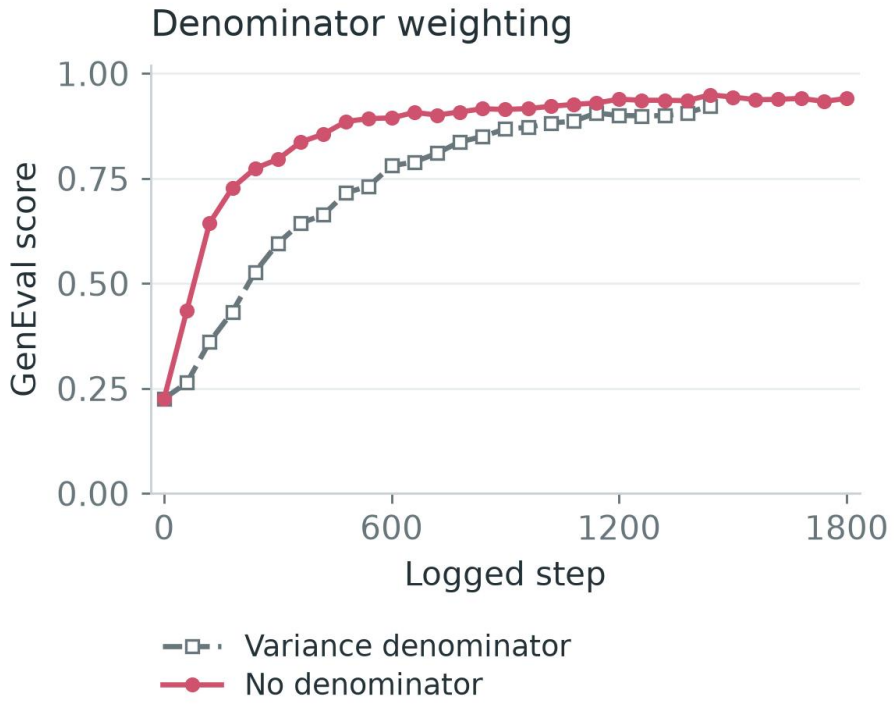
SDE exploration + training inside, ODE outside.

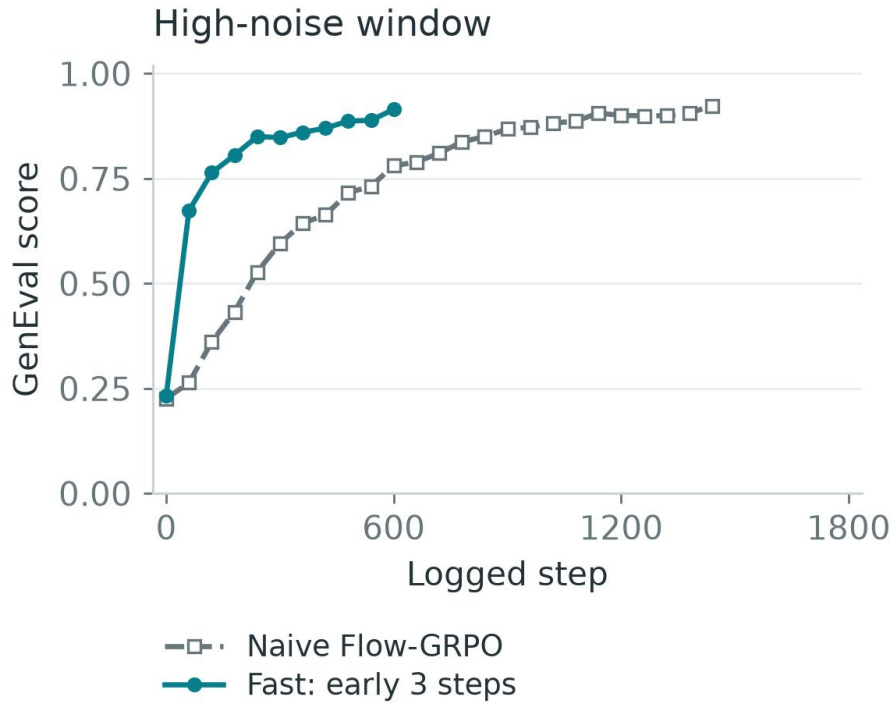
Keep group advantages and PPO clipping.

Denominator choice: soft weighting

Variance	$1/(\sigma\sqrt{\delta})$	Favors low noise
STD	1	Removes σ-scale bias
None	$\sigma\sqrt{\delta}$	Favors high noise

$f = A\delta b\xi \times (\cdot)$, current = old, with fixed δ and b.





Insight | Fewer backward passes, greater weight on actions and forces that improve reward.

3. Understanding Forward-Process RL (DiffusionNFT Family)

3.1 Preliminaries: Which Distribution Should the Model Generate?

Reweight old with goodness r ∈ [0,1] to make high-reward endpoints more likely.

$$p_+(x_0 \mid c) = \frac{r(x_0, c)}{\bar{r}_c} p_{\text{old}}(x_0 \mid c)$$

$$\bar{r}_c = \mathbb{E}_{p_{\text{old}}}[r \mid c], \qquad 0 < \bar{r}_c < 1$$

This is the desired tilted distribution (a hypothetical target).

As in Flow Matching, re-noise samples from this endpoint distribution.

$$x_t = (1-t)x_0 + t\varepsilon, \qquad u = x_0 - \varepsilon$$

$$v^+_{\text{fwd}}(s) = \mathbb{E}_{Q_+}[u \mid s], \qquad s = (x_t, t, c)$$

Q_+: the joint law of endpoints from p_+, sampled t and ε, and the noised states.

Each tilted endpoint distribution induces an ideal forward velocity field.

3.1 The Basic Idea: Learn the Ideal Field through MSE

If we could sample $x_0 \sim p_+$, we could train as in Flow Matching pretraining:

$$L_+(\theta \mid c) = \mathbb{E}_{x_0 \sim p_+,\, t,\, \varepsilon}\left[\|v_\theta(x_t, t, c) - u\|^2\right]$$

The conditional mean of the sample target u is the ideal field, so MSE is equivalent to:

$$L_+ = \mathbb{E}_{s \sim Q_+}\left[\|v_\theta(s) - v^+_{\text{fwd}}(s)\|^2\right] + \text{const}$$

$$v_\theta = v^+_{\text{fwd}} \quad \Longrightarrow \quad p_\theta(x_0 \mid c) = p_+(x_0 \mid c)$$

Ideal conditions: fit the field and integrate exactly from the same noise prior.

The obstacle: we have a sampler for p_old, but no ready-made sampler for p_+.

Before stability: pull the actual velocity field toward the ideal tilted field.

3.1 No Direct p_+ Sampler → Reweight Old Samples

Let $h\theta(x_0)$ denote the mean MSE after re-noising each endpoint:

$$h_\theta(x_0) = \mathbb{E}_{t,\varepsilon}\left[\|v_\theta(x_t, t, c) - (x_0 - \varepsilon)\|^2\right]$$

Importance weighting: change the sampling law, preserve the objective

$$\mathbb{E}_{p_+}[h_\theta] = \mathbb{E}_{p_{\text{old}}}\left[\frac{p_+(x_0 \mid c)}{p_{\text{old}}(x_0 \mid c)} h_\theta(x_0)\right]$$

$$\mathbb{E}_{p_+}[h_\theta(x_0)] = \frac{1}{\bar{r}_c}\mathbb{E}_{p_{\text{old}}}[r\, h_\theta(x_0)]$$

The basic trainable objective: goodness × MSE

$$L_{\text{basic}}(\theta \mid c) = \mathbb{E}_{x_0 \sim p_{\text{old}},\, t,\, \varepsilon}\left[r(x_0, c)\, \|v_\theta(x_t, t, c) - u\|^2\right]$$

No explicit ideal-field estimate: sample from old, re-noise, and regress with weight r.

3.1 Diffusion-NFT: Reward Centering and a Negative Branch

First convert raw group rewards into goodness (why?):

$$r_i = \frac{1}{2}\left[1 + \mathrm{clip}\left(\frac{R_i - \bar{R}_c}{Z_c}, -1, 1\right)\right]$$

NFT also defines a negative distribution favoring low-reward endpoints (why?):

$$p_-(x_0 \mid c) = \frac{1 - r(x_0, c)}{1 - \bar{r}_c}\, p_{\text{old}}(x_0 \mid c)$$

First consider on-policy: $v_c \equiv v_\theta = v_{\text{old}}$ (at the start of the update)

Interpolate around old to form two mirrored branches (why old?):

$$v_\pm = v_{\text{old}} \pm \beta\,(v_\theta - v_{\text{old}})$$

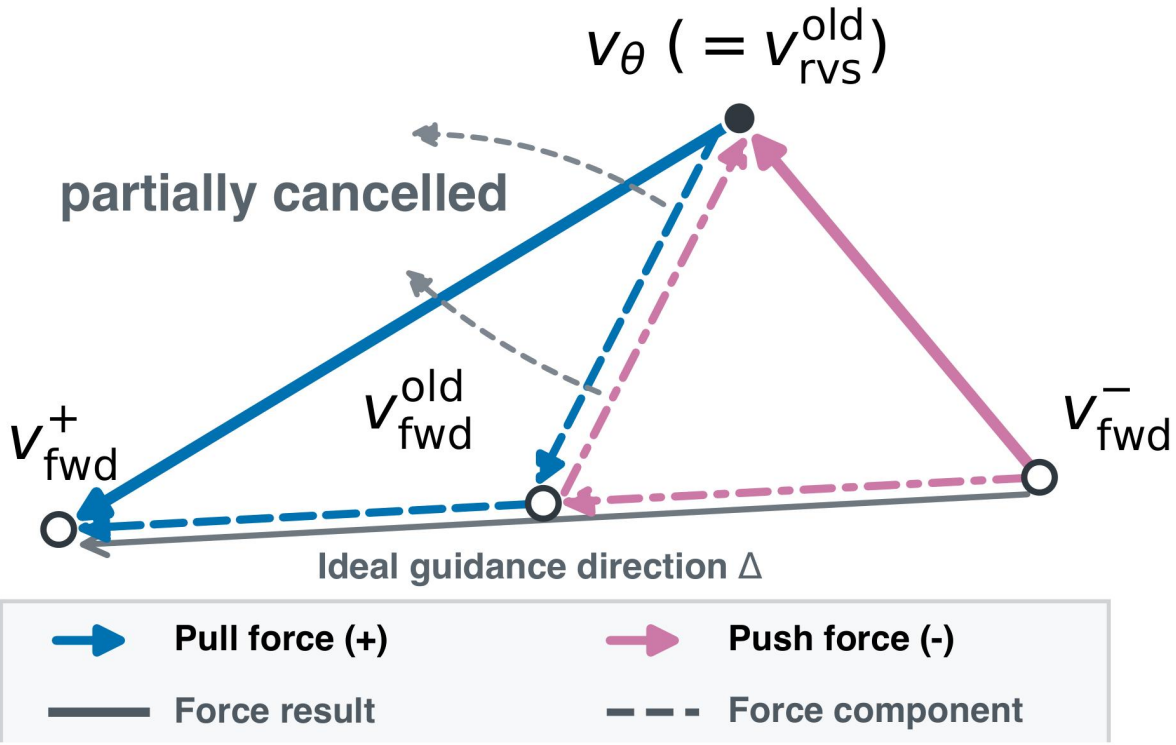
$$L_{\text{NFT}} = \mathbb{E}\left[r\|v_+ - u\|^2 + (1 - r)\|v_- - u\|^2\right]$$

Current and old have equal values, but old is frozen. Differentiate only current.

Next | What does this loss do? Why a negative branch, and why are two branches not enough?

3.2 Two Branches: Aligned Guidance, Opposing Extra Forces

At the same state s = (x_t, t, c), compare the ideal fields induced by positive, old, and negative distributions.



Lemma: Both branches give the same G

$$q = \mathbb{E}_{\text{old}}[r \mid s], \quad G = \text{Cov}_{\text{old}}(r, u \mid s)$$

Pull toward the positive field

$$q\left(v^{+}_{\text{fwd}} - v^{\text{old}}_{\text{fwd}}\right) = G$$

Push away from the negative field

$$(1-q)\left(v^{\text{old}}_{\text{fwd}} - v^{-}_{\text{fwd}}\right) = G$$

0 < q < 1. The shared positive factor 2β is omitted.

Ideal fields are collinear. Off this line, actual old adds opposing consistency forces.

First term: guidance reinforces. Second term: consistency forces seek to cancel.

3.2 Two-Branch Objective: Reward Regression and Old Anchoring

Keep the familiar MSE form: each term is a squared distance to a velocity target.

$$L_{\mathrm{NFT}} = \beta^2 \mathbb{E}_{\mathrm{old}}\left[a\|v_\theta - u\|^2 + (1-a)\|v_\theta - v_{\mathrm{old}}\|^2\right] + C$$

$$a = \frac{2r-1}{\beta}, \qquad \beta > 0$$

E_{old}: for a fixed prompt, sample $x_0 \sim p_{old}$, $t \sim \rho$, and independent Gaussian noise.

Reward-weighted regression

a > 0: pull toward sample target u

a < 0: push away from sample target u

Old anchoring

The remaining coefficient is 1–a

Together, the terms set the update direction

To separate reward guidance from consistency, introduce the ideal old forward field:

$$v^{\mathrm{old}}_{\mathrm{fwd}}(x_t, t, c) = \mathbb{E}_{\mathrm{old}}[u \mid x_t, t, c]$$

At the same state,
the conditional mean of u.

u is the sample Flow Matching velocity target. u, reward, old, and the ideal old field are frozen.

MSE is the training interface. Next, examine the forces it applies in output space.

3.2 Three Forces behind the MSE Interface

Per-sample output force $$f = -\nabla_{v_\theta} \ell_{\text{NFT}} = f_{\text{guidance}} + f_{\text{consistency}} + f_{\text{restoration}}$$

Reward guidance $$f_{\text{guidance}} = 2\beta(2r-1)\left(u - v_{\text{fwd}}^{\text{old}}\right)$$

Base direction: ideal old field → sample target u. Reversed when 2r–1 < 0.

Self-consistency $$f_{\text{consistency}} = 2\beta(2r-1)\left(v_{\text{fwd}}^{\text{old}} - v_{\text{old}}\right)$$

Base direction: actual old → ideal old field. The two branches seek to cancel.

Restoration $$f_{\text{restoration}} = 2\beta^2(v_{\text{old}} - v_\theta)$$

Current → actual old. Zero at current = old, restoring after drift.

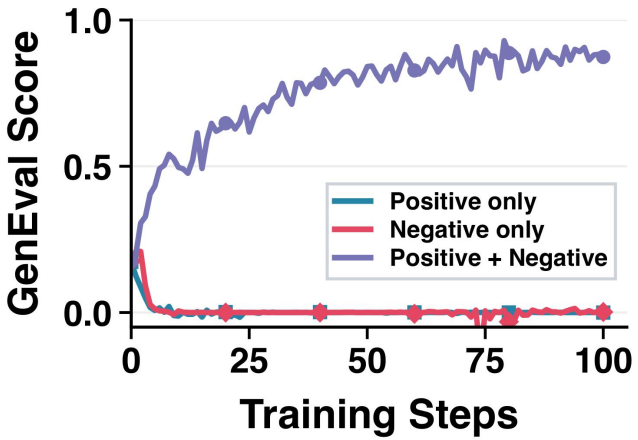
Output forces map through the Jacobian to parameter gradients, then aggregate across samples.

3.2 Why Normalize Group Rewards with Two Branches?

Balanced group coefficients ≠ zero consistency residual at every state.

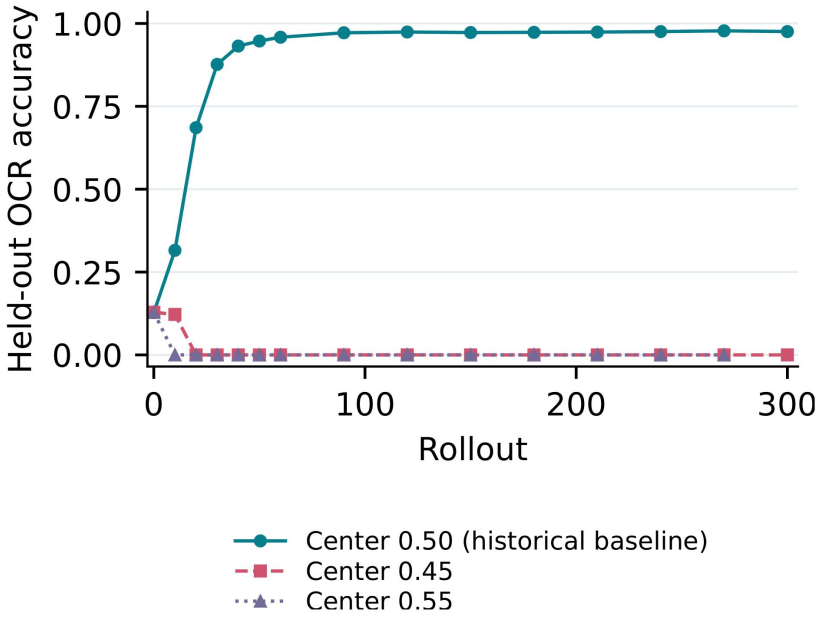
$$r_i = c_0 + \frac{1}{2}\mathrm{clip}\left(\frac{R_i - \bar{R}_c}{Z_c}, -1, 1\right)$$

Experiment 1 | Single vs. dual branch



(A)

Experiment 2 | Center shift



(B)

Shift only the goodness center

c_0 = 0.50 Sum = 0

c_0 = 0.45 –0.1K

c_0 = 0.55 +0.1K

Without clipping. K denotes group size.

Shifts in either direction cause rapid collapse.

Insight | Balanced coefficients allow cancellation, but do not ensure it in parameter gradients.

3.2 Why Introduce Frozen Old into Regression?

Direct weighted regression uses current. NFT first constructs mirrored outputs around old.

$$v_+ = \beta v_c + (1-\beta)v_{\text{old}}, \qquad v_- = -\beta v_c + (1+\beta)v_{\text{old}}$$

NFT: the target contains frozen old

$$v_{\text{tgt}} = au + (1-a)v_{\text{old}}$$

$$f = 2\beta(2r-1)(u - v_{\text{old}}) + 2\beta^2(v_{\text{old}} - v_c)$$

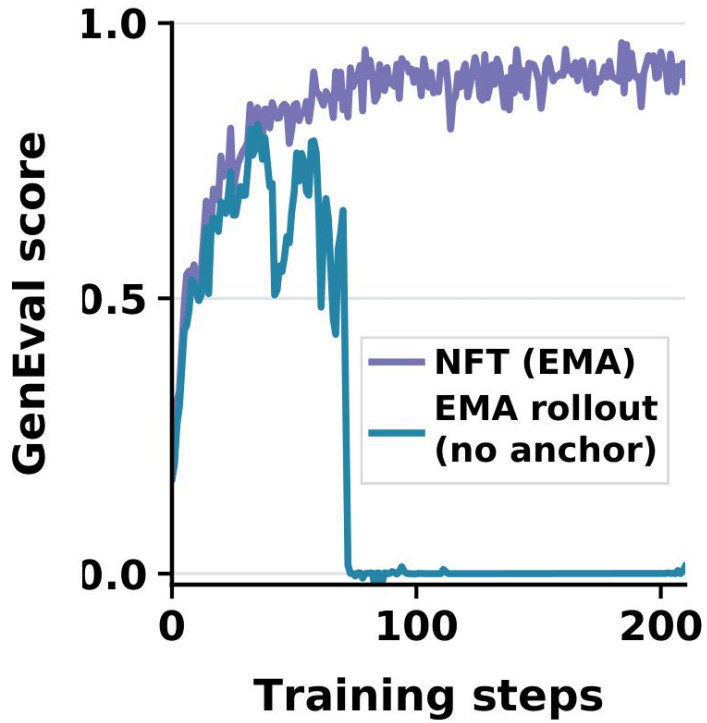
Extra restoration: $2\beta^2(v_{\text{old}} - v_c)$

Ablation: replace only old in the target with current

$$v_{\text{tgt}}^{\text{no anchor}} = au + (1-a)\text{sg}(v_c)$$

$$f_{\text{no anchor}} = 2\beta(2r-1)(u - v_c)$$

sg means stop-gradient. $v_c = v_\theta$, $a = (2r-1)/\beta$.

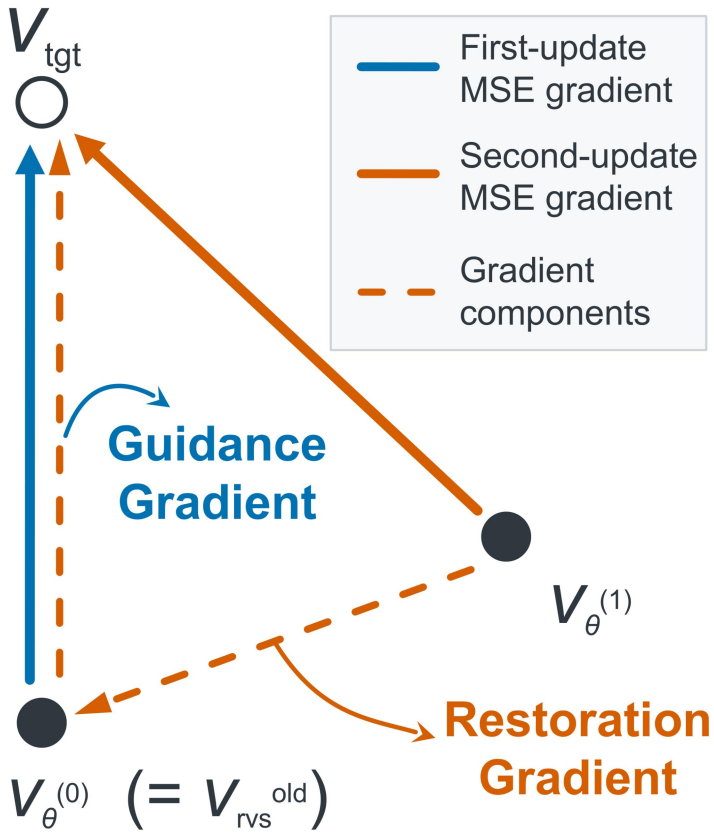


EMA rollouts alone cannot prevent collapse without the loss anchor.

Insight | Restoration comes from the old anchor in the loss, not from old rollout data.

3.2 A Second Update Creates Restoration, without EMA

One rollout batch, frozen old. Update sequentially on its two halves.



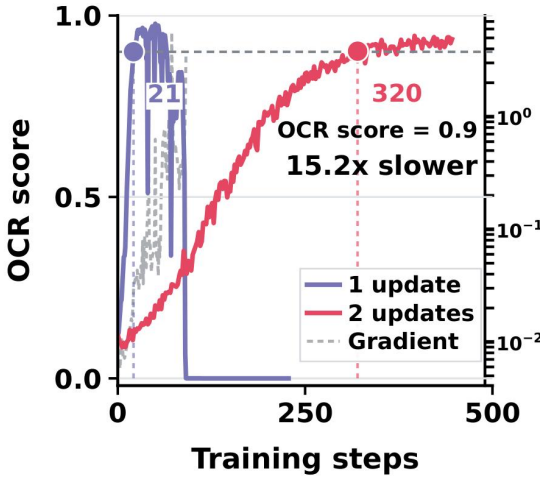
Guidance here includes reward and consistency.

First update: current starts at old

$$v_c^{(0)} = v_{\text{old}} \quad \Longrightarrow \quad f_{\text{restoration}}^{(0)} = 0$$

Second update: current has moved, old has not

$$f_{\text{restoration}}^{(1)} = 2\beta^2\left(v_{\text{old}} - v_c^{(1)}\right)$$

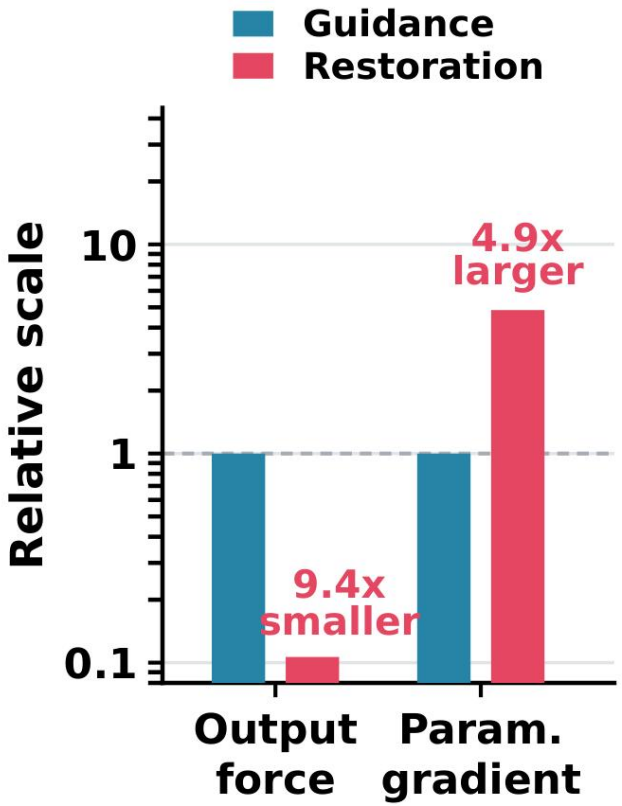


No EMA, β = 1
1 update: rapid rise, then collapse
2 updates: stable in this window

To reach OCR = 0.9:
21 vs 320 optimizer steps
About 15.2× slower

No repeated samples needed: parameter movement also creates drift at new states.

3.3 Smaller Output Forces, Stronger Parameter Gradients



Return to the three-level mental model

$$f_i \longmapsto J_i^\top f_i \longmapsto \frac{1}{N}\sum_i J_i^\top f_i$$

Force F Per-sample gradient P Aggregate gradient A

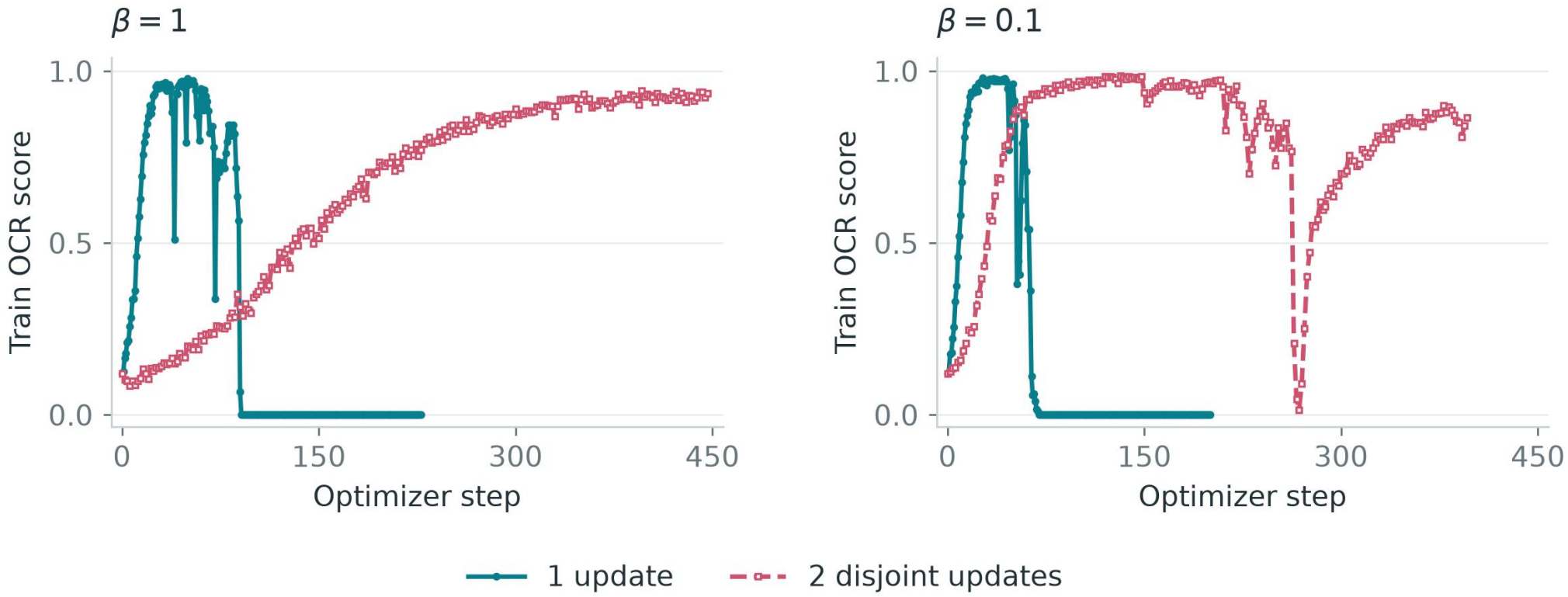
Restoration undoes changes already realized

One direction $-\Delta\theta$ can undo changes across states.
Projections on this shared return direction add up.

Insight | Two-update restoration has a high per-sample response.

3.4 NFT's β Tunes the Competition between Forces

Similar starts with one update. A second update separates learning speed and stability. EMA stays off:



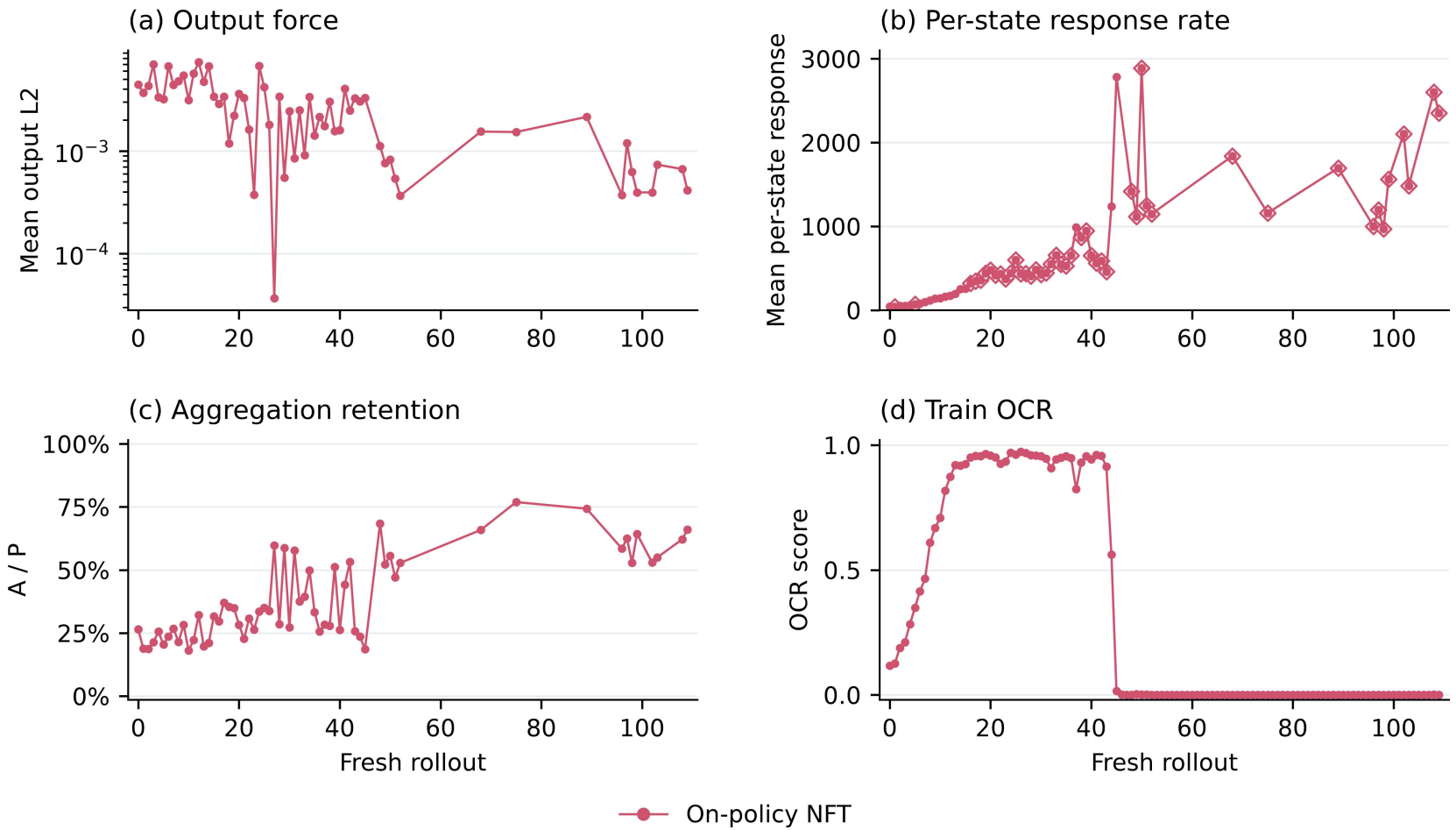
$$\frac{f}{2\beta^2} = \frac{2r-1}{\beta}(u - v_{\text{old}}) + (v_{\text{old}} - v_\theta)$$

Feedback includes reward guidance and consistency. β sets its share relative to restoration.

With restoration, β sets relative scales. EMA / two updates add off-policy competition, requiring β tuning.

3.5 NFT Instability: Small Output Forces, Spiking Response

On-policy NFT: reward collapse accompanies a spike in per-sample response.



Output force does not spike

The same output request can yield a larger per-sample gradient.

Collapse near rollout 45

Safeguards must handle parameter-space jumps, not only loss scale.

Risk lies in force-to-gradient conversion: small loss or force need not mean a small update.

3.6 EMA Tuning: Restoration Must Catch Gradient Spikes

Stability hinges on the competition between forces when a spike occurs.

$$\theta_{\text{old}} \leftarrow \rho_k \theta_{\text{old}} + (1 - \rho_k)\theta, \qquad f_{\text{rest}} = 2\beta^2(v_{\text{old}} - v_\theta)$$

Predefined restoration

EMA decay presets reference tracking speed.
β sets feedback versus restoration strength.

A sudden guidance-gradient spike

A per-sample response spike
can amplify feedback into a gradient spike.

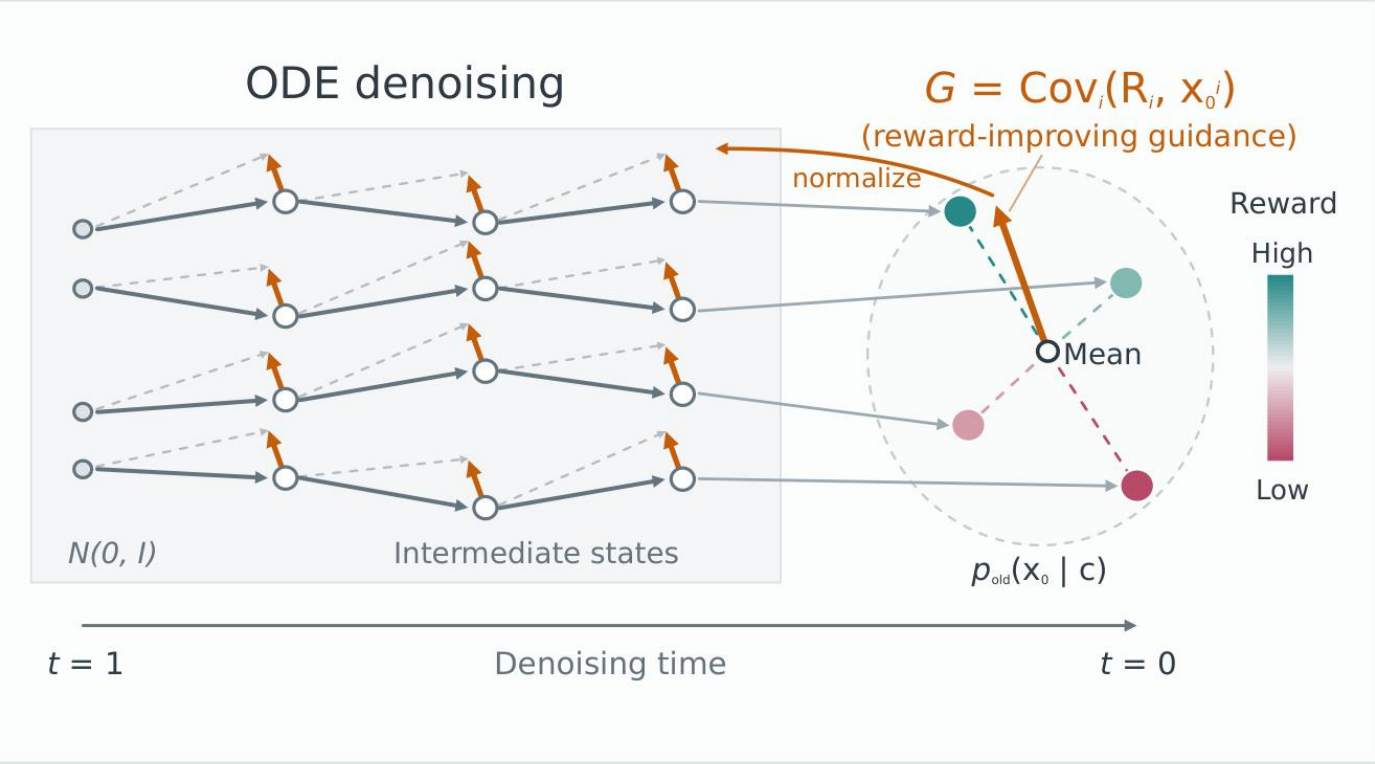
Preset restoration must oppose gradient spikes strongly enough.

The core is to stop harmful guidance updates from dominating parameter movement.

Tune for normal learning and a timely, effective restoration response to spikes.

4. Diffusion-ART: Aggregation-first Reward Tilting with Adaptive Restoration

K endpoints from each prompt jointly determine one reward-guidance direction.



Mean shift under reward tilting

$$p_\eta(x_0 \mid c) \propto p_{\text{old}}(x_0 \mid c) e^{\eta R}$$

$$\frac{\partial}{\partial \eta} \mathbb{E}_{p_\eta}[x_0 \mid c] \Big|_{\eta = 0}$$

$$= \mathrm{Cov}_{\text{old}}(R, x_0 \mid c)$$

Upweight high-reward endpoints:
the mean shifts along the covariance.

$$\widehat{g}(c) = \frac{1}{K} \sum_{i=1}^{K} (R_i - \bar{R}_c)(x_0^{(i)} - \bar{x}_{0,c}), \qquad G(c) = \frac{\widehat{g}(c)}{\mathrm{RMS}(\widehat{g}(c))}$$

Aggregate rewards in output space first, then regress with shared guidance at rollout states.

4.1 One Frozen Target, Two MSE Updates

Guidance and old stay fixed within a rollout. The second update naturally produces restoration.

Algorithm 1 DiffusionART: one training iteration

Rollout Stage

1: Sample with an ODE solver using $v_\theta^{(0)} = v_\theta$.
For each prompt c, collect K endpoints with rewards R, states s_t, and predictions $v_\theta^{(0)}(s_t)$.
2: Compute $G(c)$ by Equation 5 for each prompt c.
3: Freeze $v_{\text{tgt}}^{\text{ART}}(s_t) = v_\theta^{(0)}(s_t) + G(c)$ for each state s_t of prompt c.

Training Stage

4: Split the samples into two equal-sized disjoint minibatches B_1, B_2.
5: For each $B \in \{B_1, B_2\}$, recompute $w(s_t)$ and update θ with one step on

$$\mathcal{L}(B;\theta) = \frac{1}{|B|} \sum_{s_t \in B} w(s_t)\, \text{MSE}(v_\theta(s_t), v_{\text{tgt}}^{\text{ART}}(s_t)).$$

Frozen target

$$v_{\text{tgt}}(s) = v_{\text{old}}(s) + G(c)$$

Shared guidance + drift restoration

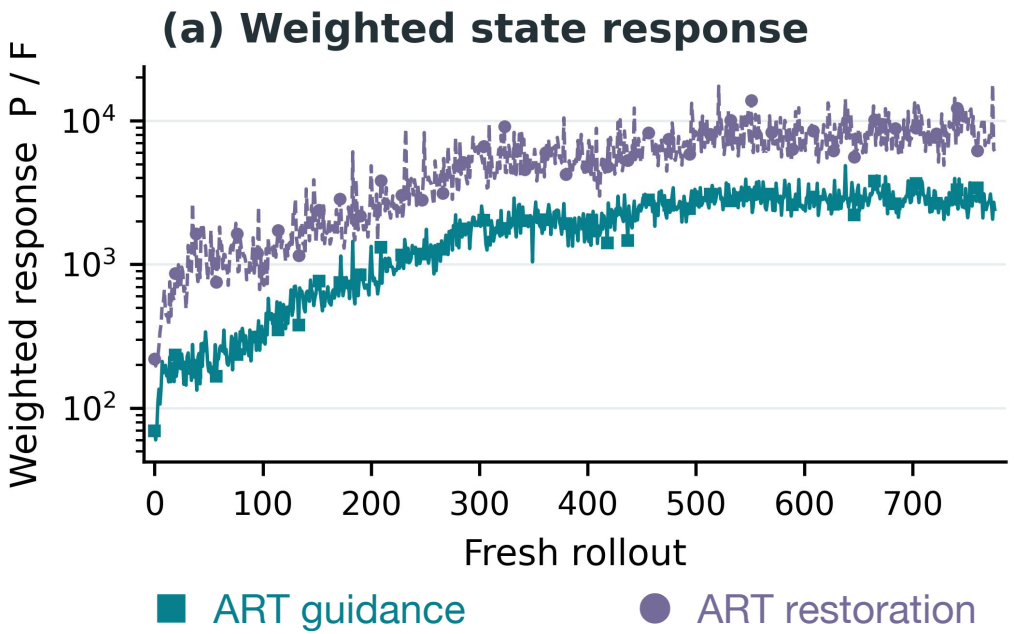
$$f(s) = \frac{2w(s)}{D}[G(c) + v_{\text{old}}(s) - v_\theta(s)]$$

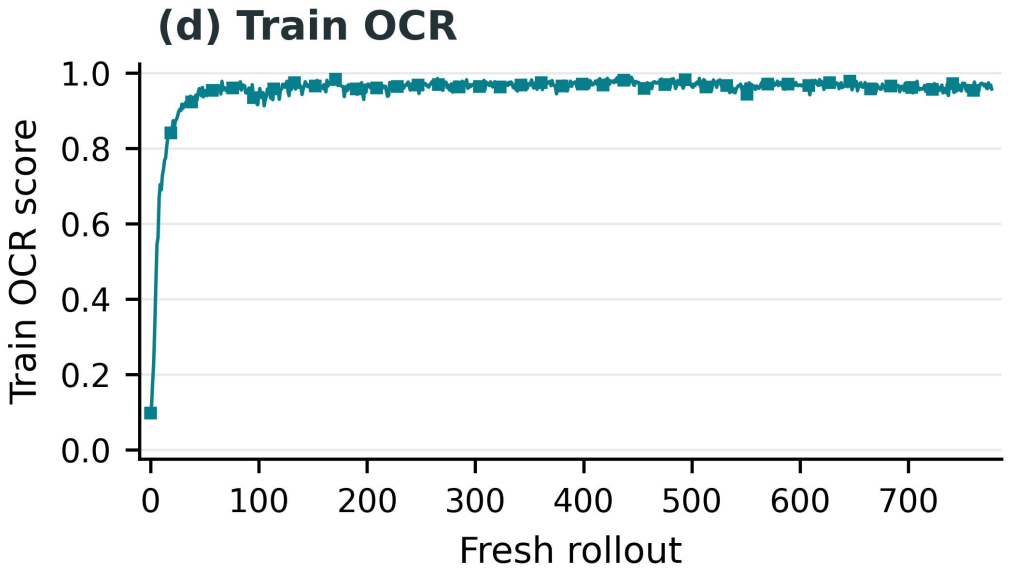
w: timestep weight + trust-region mask (block excessive outward updates, keep inward corrections). D: output dimension.

Shared guidance, frozen target. The second update restores adaptively to drift, without EMA.

4.2 ART: Guidance and Restoration Responses Grow Together

Diagnostics at the second update. Training OCR remains stable over nearly 800 observed rollouts.

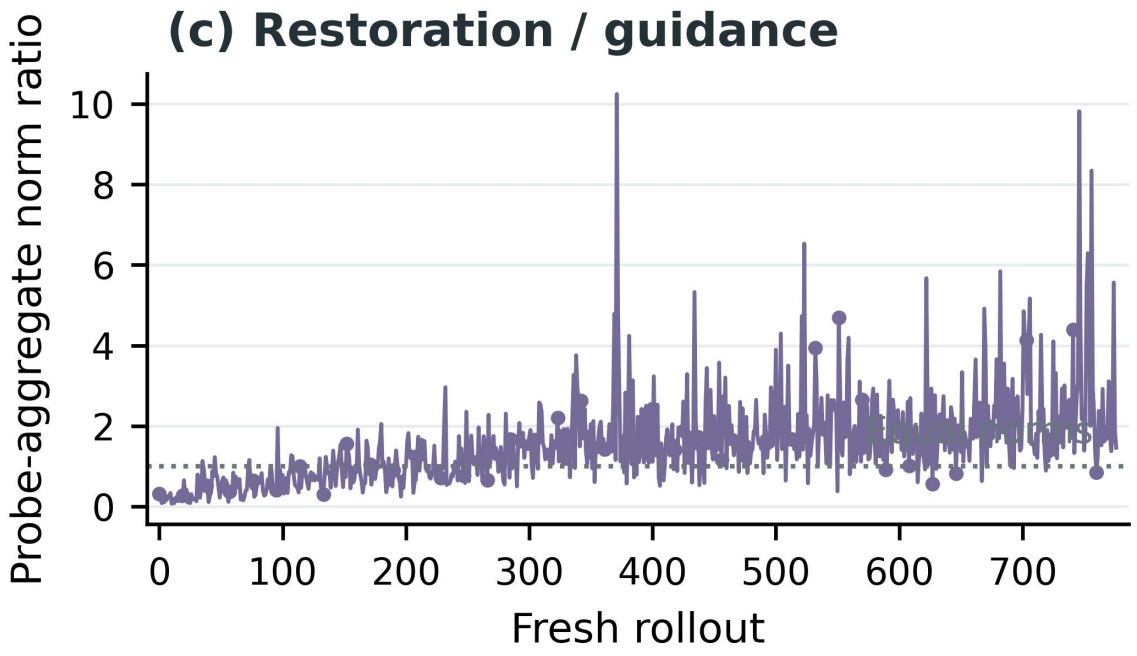




Adaptivity 1: Restoration response grows with guidance response and consistently exceeds it.

4.3 Adaptive Restoration Responds to Actual Drift

A shared frozen target automatically gives the second update a restoring signal matched to current drift.



First update: actual drift

$$\Delta v(s) = v_{\theta^{(1)}}(s) - v_{\text{old}}(s)$$

Second update: restoration

$$f^{(1)}_{\text{rest}}(s) = -\frac{2w(s)}{D}\Delta v(s)$$

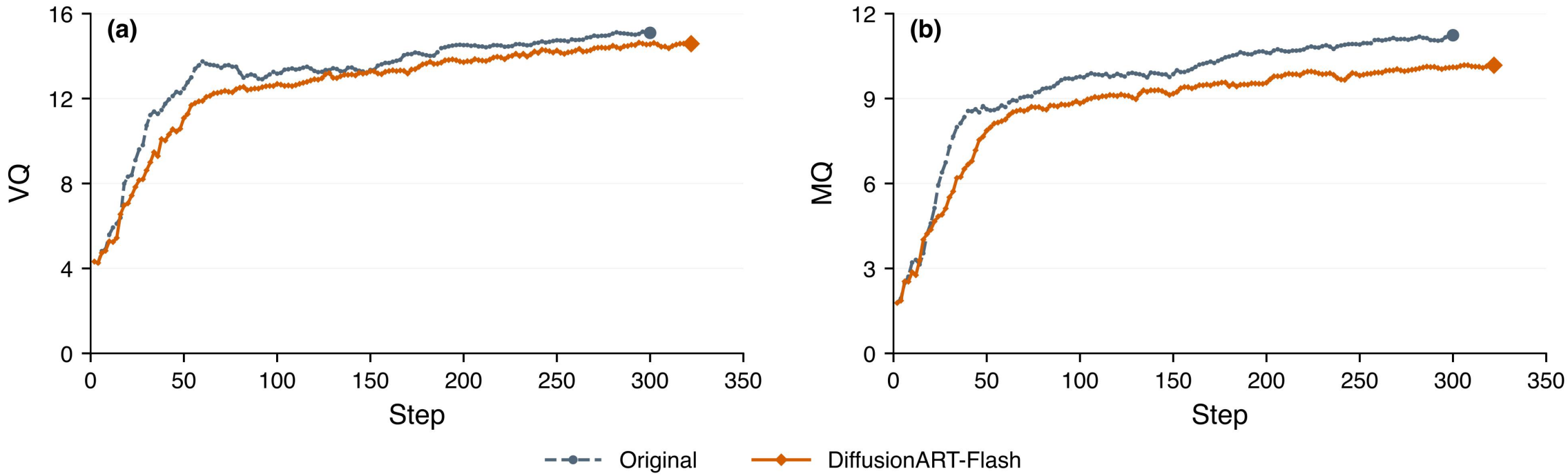
Sensitive directions drive the first move.
The resulting drift can also be sensitive.
This agrees with response co-growth.

Adaptivity 2: Restoration strength follows actual drift, with no preset fixed lag.

4.4 DiffusionART-Flash

Train two high-noise positions while retaining the complete rollout

16-step HunyuanVideo rollout. Randomly select 2 of the first 4 positions.



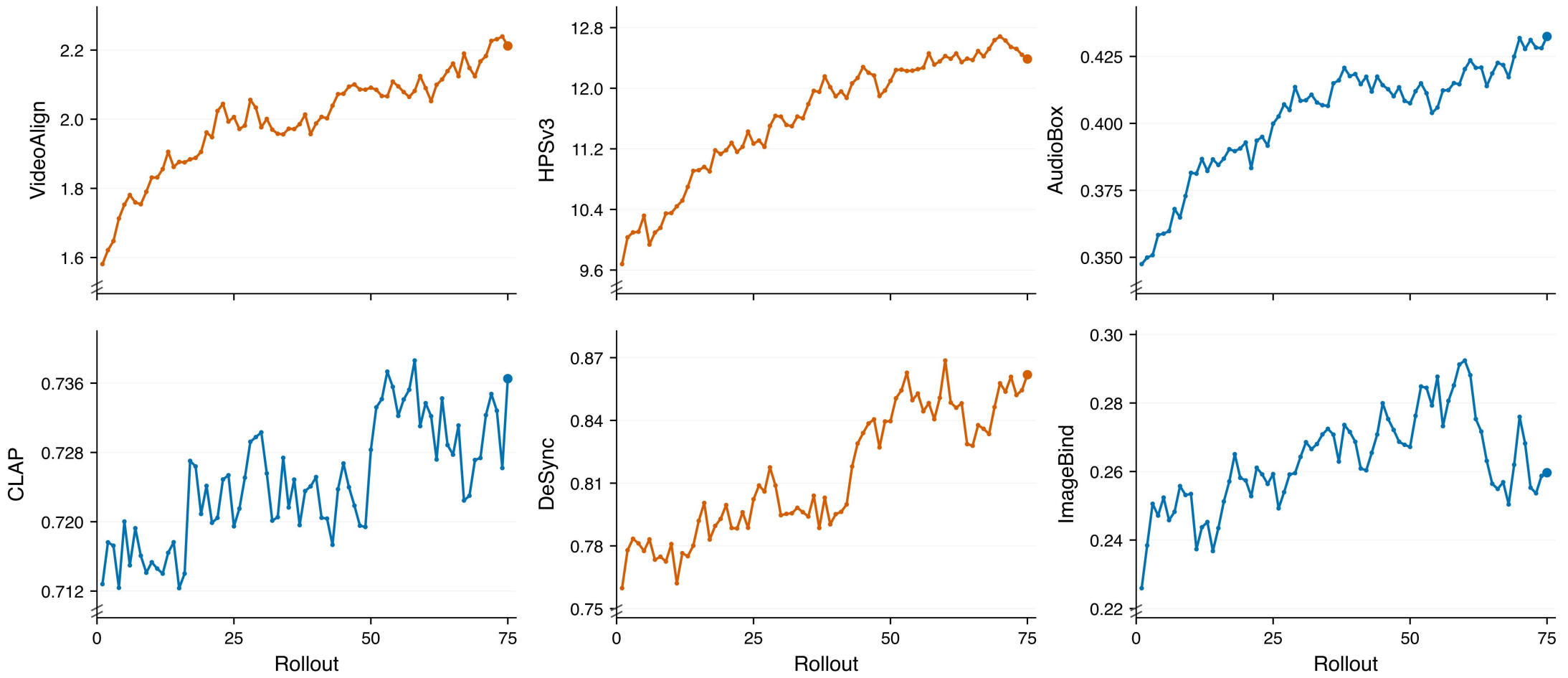
VQ:MQ = 1:1. Raw training scores. Keep both disjoint updates and average over the selected pairs.

Comparable VQ and MQ with 8× fewer update pairs than training all 16 positions

4.5 DiffusionART-Flash on MiniMax-H3

Full-parameter audio/video training at 480P, ~5 seconds, with 10-step rollouts

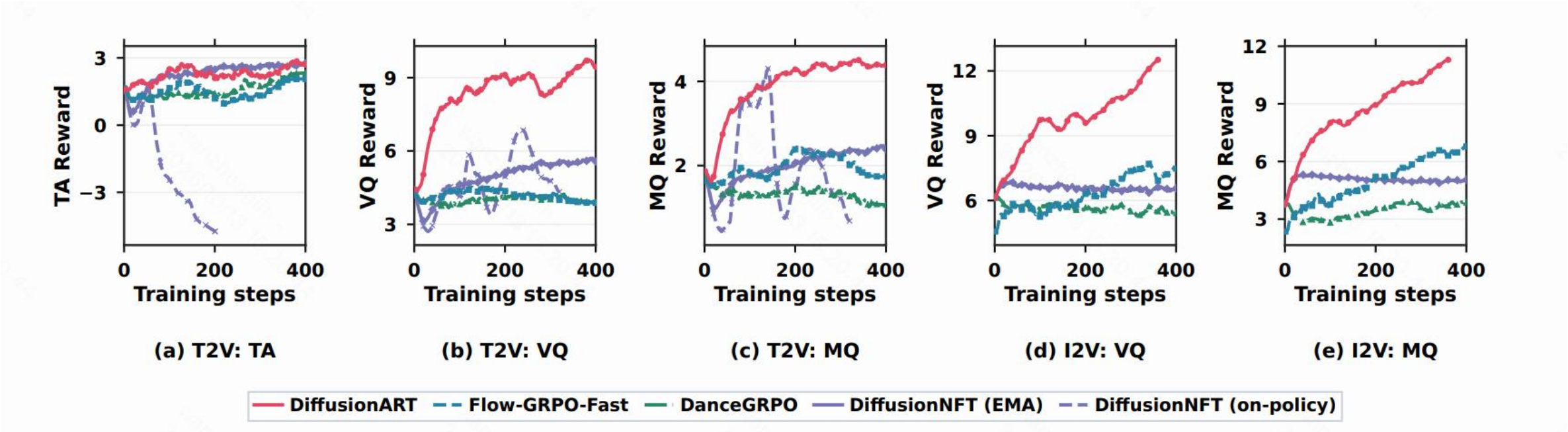
Video: 2 of the first 4 positions. Audio: 2 of the first 3 positions.



Six raw reward components before prompt-group normalization. 75 rollouts = 150 updates.

5.1 Video performance

HunyuanVideo-13B (T2V) and Wan2.1-14B (I2V)



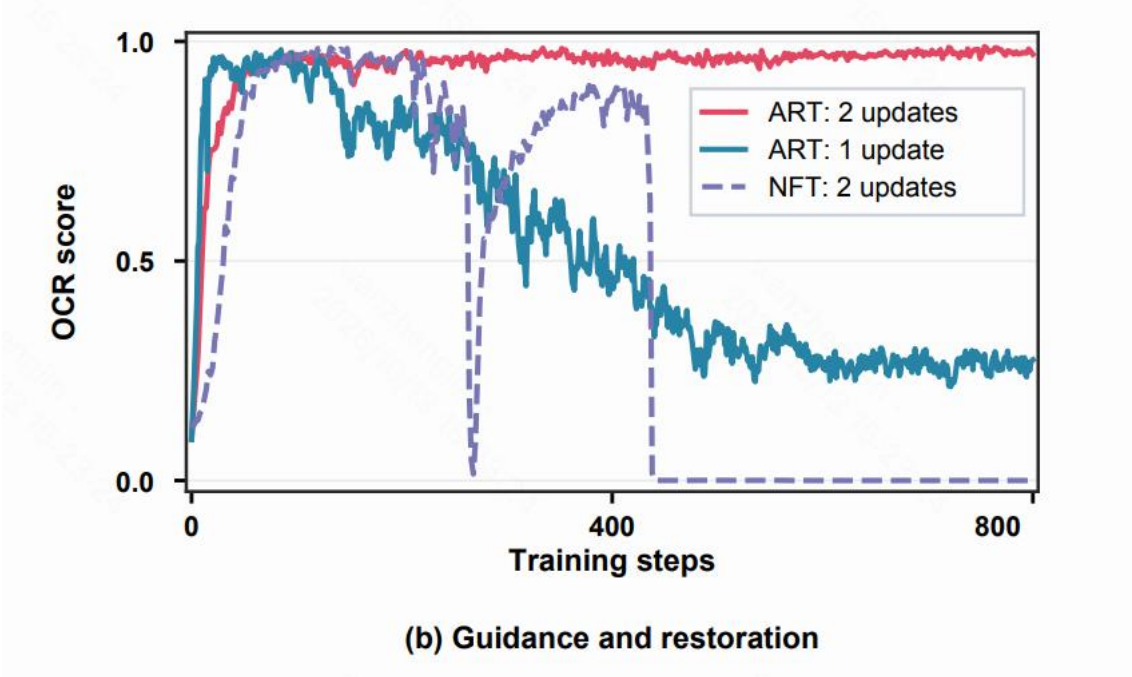
Selected final evaluation results (higher is better)

Method	T2V VQ	T2V MQ	T2V TA	I2V VQ	I2V MQ
DiffusionART	**8.970**	**4.224**	**2.111**	**11.196**	**10.476**
DiffusionNFT (EMA)	6.827	2.683	2.096	5.516	4.161
Flow-GRPO-Fast	3.149	1.874	1.667	5.771	4.710

5.2 Image ablation, pairwise feedback, and small K

Guidance and restoration matter. Pairwise feedback works with only two samples

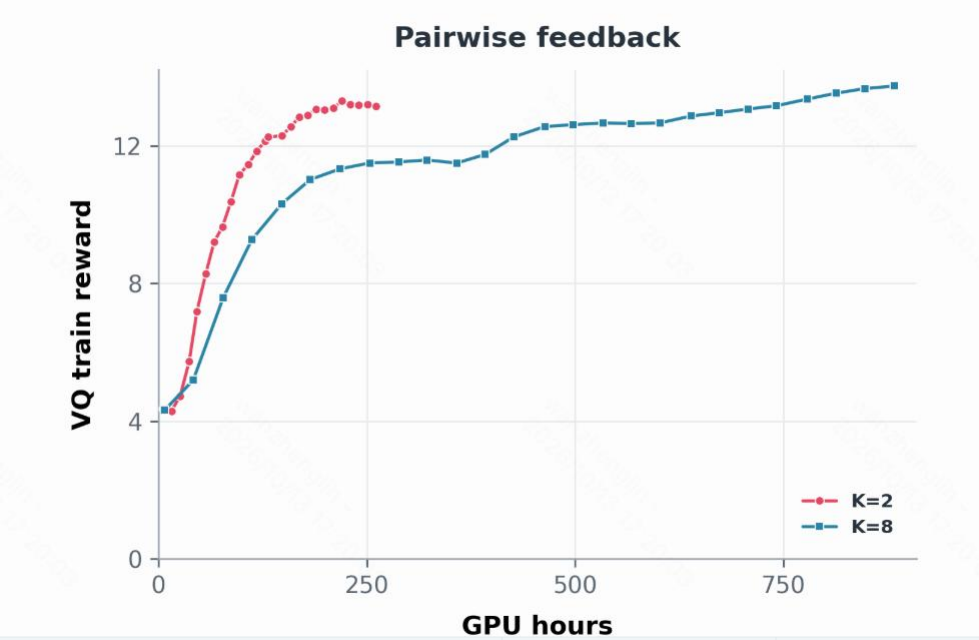
Image ablation: OCR



(b) Guidance and restoration

ART with two updates remains stable.
One ART update loses restoration.
Two NFT updates still collapse.

Video: pairwise feedback



Feedback	Eval VQ	GPU hours
Pairwise K=2	**12.235**	**260.8**
Pairwise K=4	10.883	465.6
Pairwise K=8	12.092	883.3
Scalar K=8	13.637	891.1

Pairwise K=2 reaches comparable VQ to K=8 with 3.4× fewer GPU hours

References and resources

Flow-GRPO: Training Flow Matching Models via Online RL

Jie Liu, Gongye Liu, Jiajun Liang, Yangguang Li, Jiaheng Liu, Xintao Wang, Pengfei Wan, Di Zhang, and Wanli Ouyang. 2025.

arxiv.org/abs/2505.05470

DiffusionNFT: Online Diffusion Reinforcement with Forward Process

Kaiwen Zheng, Huayu Chen, Haotian Ye, Haoxiang Wang, Qinsheng Zhang, Kai Jiang, Hang Su, Stefano Ermon, Jun Zhu, and Ming-Yu Liu. 2025.

arxiv.org/abs/2509.16117

DiffusionART: Efficient, Stable and Scalable Reinforcement of Diffusion Models

Zhenglin Wan, Jiayu Chen, Zhengfeiyang Zhang, Pengfei Zhou, Wangbo Zhao, Fangyuan Kong, Pengfei Wan, Kun Gai, and Yang You. 2026.

arxiv.org/abs/2610.12572

Code and recipes

github.com/vanzll/DiffusionART

Blog (Insight Report Coming Soon to Arxiv)

vanzll.github.io/blog/diffusion-rl/