Lyapunov-Guided Self-Alignment: Test-time Adaptation for Offline Safe Reinforcement Learning

Seungyub Han* Hyungjin Kim* Jungwoo Lee (* Equal Contribution)

Seoul National University

Project page
(paper & code)

1. Motivation

Offline RL agents are pretrained on large curated datasets (e.g. D4RL, DSRL), but **distribution shift at deployment** drives them into unsafe regions. Existing remedies either

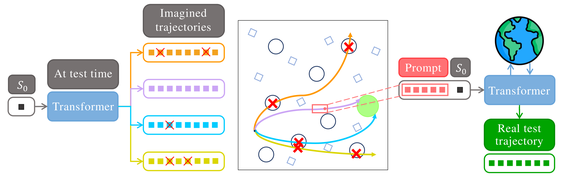
- **Retrain** with uncertainty/constrained objectives (belief-based adaptation), or
- Add **cost-sensitive** signals that require fine-tuning and sacrifice return.

Can we align a *pretrained* offline RL agent for safety *at test time* – with *no* parameter updates?

2. Key Idea: Self-Alignment from LLMs → RL

Recast the LLM recipe of *self-alignment* (Sun et al., 2023) for RL:

- Pretrained Decision Transformer + VAE world model **imagines** many rollouts
- **Filters** them by a Lyapunov condition
- **Recycles** the safest segment as an in-context **prompt**



From a fixed initial state, the transformer imagines multiple rollouts, flags risky state–action pairs with the Lyapunov condition (✖), and extracts a safe segment as a prompt to steer the real trajectory (hazards: black/blue markers; goal: green).

Analogy: LLM prompt ↔ demonstration trajectory; latent concept θ ↔ high-level policy parameter; aligned output ↔ safe trajectory.

3. Contributions

- **Lyapunov stability** formulation for offline RL as an **occupancy-measure** criterion.
- **Hierarchical-RL / Bayesian** interpretation: prompting = inference over latent skills.
- **SAS**: Lyapunov-guided prompts for safe test-time adaptation – *no retraining, no cost data.*
- Up to 2× lower cost & failure improving return across Safety Gymnasium, Bullet-Safety-Gym.
- **Formal safety bound**: probabilistic guarantee with moderate compute (Thm 2).

4. Setting: CMDP + Hierarchical Policy

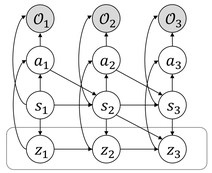
Discounted CMDP $\langle \mathcal{S}, \mathcal{A}, r, c, \mathcal{P}, p_1, \gamma \rangle$ with per-step cost bounded by $C_{\max}$.

Hierarchical policy (high-level over skills z_t + low-level):

$$\pi(a_t \mid s_t) \;=\; \sum_{z_t} \underbrace{\pi_\theta^{\text{high}}(z_t \mid s_t, z_{t-1})}_{\text{high-level}} \cdot \underbrace{\pi_\phi^{\text{low}}(a_t \mid s_t, z_t)}_{\text{low-level}}.$$

Safe RL objective:

$$\max_\pi \; J_R(\pi) \;\text{ s.t. }\; J_C(\pi) \le d,$$
$$\text{where } \; J_R(\pi) = \mathbb{E}_{(s,a)\sim\rho^\pi}[r(s,a)], \quad J_C(\pi) = \mathbb{E}_{(s,a)\sim\rho^\pi}[c(s,a)].$$



HRL as a PGM: latent skills z_t (HMM); optimality/Lyapunov observables $\mathcal{O}_\tau, \mathcal{U}_t, \mathcal{V}_t$ plug in as binary evidence.

5. Model-Based Occupancy Measure

DT+VAE world model gives an estimated transition $\hat{\mathcal{P}}$, hence

$$\hat{\rho}^\pi(s,a) = \sum_{t=1}^{\infty} \gamma^{t-1}\, \mathbb{P}_{\pi,\hat{\mathcal{P}}}(s_t = s, a_t = a \mid s_1 \sim p_1).$$

Energy (negative log-density): $E(s,a) \;=\; -\log \hat{\rho}^\pi(s,a)$.

With concentrability $D_{\text{conc}} \ge 1$ (offline coverage), the safe policy must stay in high-density regions; equivalently, in low-energy regions:

$$\mathcal{R}_\rho = \Big\{(s,a) : \hat{\rho}^\pi(s,a) \ge \frac{d}{C_{\max} D_{\text{conc}}}\Big\} \;\equiv\; \mathcal{R}_E = \Big\{(s,a) : E(s,a) \le -\log \frac{d}{C_{\max} D_{\text{conc}}}\Big\}.$$

Takeaway: *safe = in-support*; energy gives the same set in log-scale.

6. Lyapunov Density Model (LDM) Recap

LDM (Kang et al., 2022) learns $G : \mathcal{S}\times\mathcal{A} \to \mathbb{R}_{\ge 0}$ with

$$G(s_e, a_e) = 0, \quad G(s,a) > 0 \text{ otherwise,}$$
$$G(s,a) \;\ge\; \gamma\, \mathbb{E}_{s'\sim\mathcal{P}}\big[\min_{a'} G(s',a')\big].$$

G is *non-increasing* along feasible trajectories ⇒ $\mathcal{R}_c = \{G \le c\}$ is **control-invariant**.

LDM backup:

$$(\mathcal{T}_{\text{LDM}} G)(s,a) = \max\{E(s,a),\, \gamma\, \mathbb{E}_{s'} \min_{a'} G(s',a')\}.$$

Test-time policy: since smaller G means safer (closer to $\{G=0\}$), the agent selects $a^*(s) = \arg\min_a G(s,a)$ on the learned G.

Caveat: this safety guarantee *requires additionally training the pretrained model with the LDM backup* to converge G – not a free test-time fix.

7. Our Lyapunov Model G_{SAS}

Replace $-\log \hat{\rho}^\pi$ with the *worst-case minimum* along a trajectory ⇒ enforces Lyapunov decrease and matches $\mathcal{R}_E$ from block 5:

$$G_{\text{SAS}}(s,a) \;=\; \log \hat{\rho}^\pi(s,a) \;-\; \underbrace{\max_\tau \min_{t'} \log \hat{\rho}^\pi(s_{t'}, \pi(s_{t'}))}_{\text{worst-case along policy}}$$

$G_{\text{SAS}} \le 0$ on $\mathcal{R}_E$ and *non-increasing* along feasible trajectories ⇒ $\mathcal{R}_G^{\text{SAS}} = \{G_{\text{SAS}} \le 0\}$ is **control-invariant** (a tighter version of $\mathcal{R}_E$).

Two implications:

- *Conservative behavior* – policy stays in high-density regions, avoiding OOD actions.
- *Test-time self-alignment* – G_{SAS} is a prior that guides prompt selection.

8. Lyapunov Condition as Probabilistic Inference

Introduce two binary observables along a trajectory:

$$\mathcal{U}_t = \mathbf{1}\{G_{\text{SAS}}(s_t,a_t) > 0\}, \quad \mathcal{V}_t = \mathbf{1}\{G_{\text{SAS}}(s_t,a_t) - G_{\text{SAS}}(s_{t+1},a_{t+1}) \ge 0\}.$$

$\mathcal{U}_t$: energy inside the invariant set. $\mathcal{V}_t$: Lyapunov descent one step forward.

Theorem 1 (Lyapunov ≡ Log-likelihood)

Up to an additive constant, G_{SAS} is equivalent to

$$\max_\tau \; \tfrac{1}{T} \sum \big[\log P(\mathcal{U}_t = 1 \mid \tau) + \log P(\mathcal{V}_t = 1 \mid \mathcal{U}_t = 1, \tau)\big].$$

⇒ safety verification reduces to **likelihood maximisation over imagined trajectories**, natively done by a transformer's In-Context Learning (ICL).

9. Prompting = Bayesian Inference over Skills

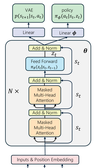
Extending implicit Bayesian ICL (Xie et al., 2021) with low-level policy π_ϕ^{low} and dynamics $\hat{\mathcal{P}}$:

$$p(\mathcal{O}_{\text{traj}} = 1 \mid \mathbf{p}_{1:K}, s_1^{\text{test}}) = \int_\theta \sum_{z_1} g_{\pi_\theta}(\tau \mid z_1, s_1^{\text{test}}) \prod_{t=1}^{T} p(\mathcal{O}_t = 1 \mid s_t, a_t)\, e^{K\, r_K(\theta)}\, p(\theta)\, d\theta.$$

Prompt $\mathbf{p}_{1:K}$ = *posterior-shaping* device: a high-likelihood prefix picks a safe skill π_θ^{high} already in the pretrained weights – **no gradient updates**, just LLM-style ICL for RL skills.

10. SAS Prompt Generation on DT + VAE

Input: pretrained π_θ, init. state s_1^{test}, prompt length K, budgets N, M.
Stage 1 — enforce $\mathcal{U}_t$ (energy upper bound)
1: for $i = 1, \ldots, N$ **do**
  imagine $\tau^{(i)} \sim p(\tau \mid s_1^{\text{test}})$
  record $\hat{E}_i = \max_t E_t(\tau^{(i)})$, $t_i := \arg\max_t E_t$
  end for
2: $i^* \leftarrow \arg\min_i \hat{E}_i$; $\mathbf{p}_{1:K} \leftarrow \tau^{(i^*)}[t_{i^*} - K + 1 : t_{i^*}]$
Stage 2 — enforce $\mathcal{V}_t$ (Lyapunov descent)
3: for $j = 1, \ldots, M$ **do**
  imagine $\tau^{(j)} \sim p(\tau \mid \mathbf{p}_{1:K}, s_1^{\text{test}})$
  score $v_j = \sum_t \mathcal{V}_t(\tau^{(j)})$
  end for
4: $j^* \leftarrow \arg\max_j v_j$; $\mathbf{p}_{1:K} \leftarrow \tau^{(j^*)}[t_{j^*} - K + 1 : t_{j^*}]$
Output: prompt $\mathbf{p}_{1:K}$; deploy $\pi_\theta(\cdot \mid \mathbf{p}_{1:K}, s_1^{\text{test}})$



Algorithm (left): two-stage rejection sampling — Stage 1 picks the prefix with the lowest peak energy ($\{\mathcal{U}_t\}$); Stage 2 re-samples from that prefix and keeps the trajectory with the largest cumulative Lyapunov descent ($\{\mathcal{V}_t\}$).
Architecture (right): DT + VAE decoder makes the transformer generative in state space ⇒ the model can **imagine** rollouts without a simulator. **High-level** (θ): FFN head → latent skill z_t (π_θ^{high}); **Low-level** (ϕ): attention+residual over (s_t, z_t), linear head → a_t (π_ϕ^{low}); **World model:** linear+VAE decoder predicts s_{t+1}.

11. Probabilistic Safety Guarantee

Assumption. $\|G_{\text{SAS}}(s_t,a_t) - G_{\text{SAS}}(s_{t+1},a_{t+1})\| \le L$ (per-step Lipschitz of G_{SAS} along trajectories).

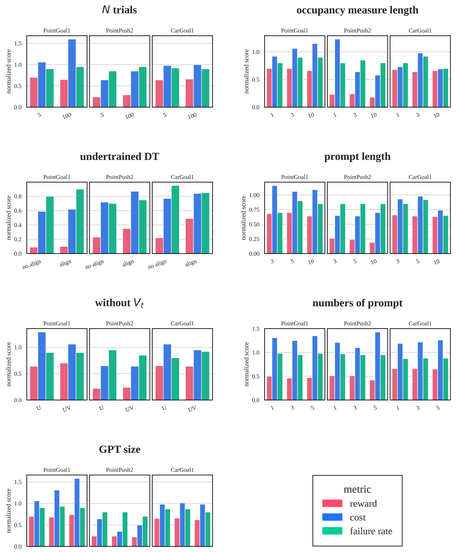
Theorem 2 (probabilistic safety)

For τ_{best} returned by Alg. 1,

$$\mathbb{P}[\tau_{\text{best}} \not\subset \mathcal{R}_G^{\text{SAS}}] \le \underbrace{\Big(\frac{\mathbb{E}_p[-\log \hat{\rho}]}{c_2}\Big)^{NT}}_{\text{Stage 1: Markov}} + \underbrace{\exp\Big(-\frac{2M\kappa^2(c_2-c_1)^2}{TL^2}\Big)}_{\text{Stage 2: Hoeffding}}.$$

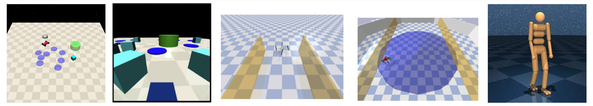
Larger N, T tighten the energy check; larger M tightens the descent check.

12. Ablation Studies



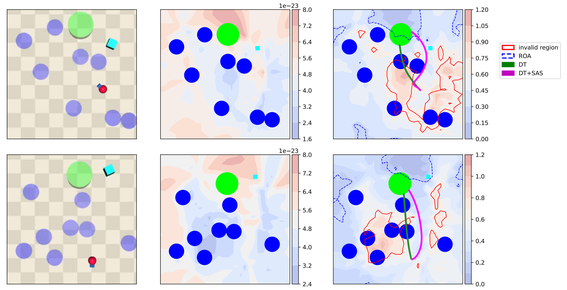
Theory checks. *Without* $\mathcal{V}_t$ (Lyapunov stage off) → cost & failure jump ⇒ Hoeffding term of Thm 2. *Undertrained DT*: align lifts reward, cuts failure on half-trained DT ⇒ SAS adds safety.
Robustness. N∈{100}, T=3 decisive, K∈{3, 5, 10} tied; ϕ prompts 1 strongest; GPT size: larger → fewer failures. ⇒ *no benchmark-specific tuning*.

13. Experiments – Environments



Safety Gymnasium Agent view of PointGoal Run Circle MuJoCo

14. Why is SAS safer?



Left: A Point agent navigates from a fixed start to a green goal while avoiding 8 fixed blue hazards and one movable cyan *vase*.
Middle: Density $\hat{\rho}^\pi(\mathbf{s})$ learned by the transformer-based agent at each point.
Right: Lyapunov map G_{SAS}.

- **Red curve**: invalid region, exceeding the 95-th percentile of G_{SAS}.
- **Blue curve**: ROA, bounded by the 10-th percentile of G_{SAS}.

Benchmark Results

Safety-Gymnasium & Bullet-Safety-Gym *(aggregate over tasks)*

	Safety-Gymnasium (12 tasks)			Bullet-Safety-Gym (8 tasks)		
Method	R↑	C↓	Safe?	R↑	C↓	Safe?
CDT+SAS (ours)	**0.392**	**0.895**	✓	**0.34**	**0.37**	✓
DT+SAS (ours)	0.402	1.03	–	–	–	–
CDT	0.389	1.18	✗	0.68	1.04	✗
BC-Safe	0.151	0.602	✓	0.52	0.82	✓
DCRL	0.098	0.737	✓	–	–	–
BCQ-Lag	0.319	1.52	✗	0.74	3.11	✗
BEAR-Lag	–	–	–	0.49	4.19	✗
CPQ	0.416	3.63	✗	0.27	1.77	✗
COptDICE	0.172	1.23	✗	0.55	3.90	✗

Safety-Gymnasium, per-task *(Prompt-selection ablation)*

Method		PG1	PG2	PP1	PP2	PB1	PB2	CG1	CG2	CP1	CP2	CB1	CB2	**Avg**
DT	R	0.660	0.377	0.218	0.202	0.379	0.495	0.638	0.513	0.350	0.204	0.237	0.212	0.374
	C	1.319	2.625	0.927	0.782	**1.188**	1.309	0.976	1.466	0.678	1.174	1.419	1.045	1.242
	F	0.883	1.000	0.667	0.875	0.950	0.983	0.917	0.925	0.667	0.950	0.950	0.950	0.893
DT+SAS (ours)	R	0.655	0.650	0.283	0.242	**0.485**	0.508	0.666	0.483	0.307	0.218	0.174	0.138	**0.401**
	C	1.185	**1.783**	**0.622**	**0.639**	1.375	1.205	**0.846**	**1.148**	**0.513**	**1.158**	**1.083**	**0.836**	**1.033**
	F	**0.867**	**0.983**	0.767	**0.850**	0.950	**0.967**	**0.867**	**0.850**	**0.483**	**0.900**	0.975	1.000	**0.872**
DT+rand	R	0.665	0.587	0.303	0.240	0.445	0.462	0.672	0.507	0.311	0.230	0.175	0.111	0.392
	C	1.258	1.811	0.678	0.758	1.485	**0.960**	1.002	1.438	0.549	1.341	1.259	0.963	1.125
	F	0.900	1.000	0.767	0.875	1.000	0.950	0.867	0.975	0.617	0.950	0.975	0.925	0.900
DT+maxmax	R	0.644	0.521	0.271	0.200	0.486	0.441	0.636	0.512	0.321	0.206	0.131	0.126	0.375
	C	**0.990**	2.152	0.640	0.730	1.808	1.273	1.034	1.497	0.574	1.271	1.103	0.911	1.165
	F	0.775	1.000	0.767	0.783	1.000	0.950	0.817	0.933	0.625	0.975	0.967	0.975	0.881

Mujoco D4RL

	Hopper		Walker2d		Humanoid	
	Reward↑	Failure Rate↓	Reward	Failure Rate	Reward	Failure Rate
DT	86.6	1.00	82.2	0.54	40.5	0.97
DT+SAS	**87.5**	**1.00**	**89.5**	**0.46**	**50.6**	**0.87**

Task	*CQL (ref.)*	DT	**DT+SAS (ours)**	SAC-N	APE-V
hopper-medium-expert	96.9	111.8	**110.4**	110.0	105.7
hopper-medium-replay	86.3	94.3	**97.3**	101.8	98.5
walker2d-medium-expert	109.1	108.3	**107.5**	116.0	110.0
walker2d-medium-replay	76.8	43.9	**69.1**	78.7	82.9

Δ *DT→DT+SAS*: −1.3%, +3.2%, −0.7%, +57.4% | Δ *SAC-N→APE-V*: −3.9%, −3.2%, −5.2%, +5.3%