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

Seungyub Han * **Hyungjin Kim*** **Jungwoo Lee**†
Seoul National University

Abstract

Offline reinforcement learning (RL) agents often fail when deployed, as the gap between training datasets and real environments leads to unsafe behavior. To address this, we present SAS (Self-Alignment for Safety), a transformer-based framework that enables test-time adaptation in offline safe RL without retraining. In SAS, the main mechanism is self-alignment: at test time, the pretrained agent generates several imagined trajectories and selects those satisfying the Lyapunov condition. These feasible segments are then recycled as in-context prompts, allowing the agent to realign its behavior toward safety while avoiding parameter updates. In effect, SAS turns Lyapunov-guided imagination into control-invariant prompts, and its transformer architecture admits a hierarchical RL interpretation where prompting functions as Bayesian inference over latent skills. Across Safety Gymnasium and MuJoCo benchmarks, SAS consistently reduces cost and failure while maintaining or improving return.

1 INTRODUCTION

Ensuring safety is a fundamental challenge for deploying RL in the real world. Recent advances in deep RL algorithms (Haarnoja et al., 2018; Janner et al., 2019; Lee et al., 2023; Eysenbach et al., 2022) have shown remarkable performance in simulation, but translating these gains to practice remains difficult, especially when a downstream controller must operate underactuated robots (Tedrake, 2009) or uncertain environments. Even small distribution shifts between training and deployment can cause highly unsafe behaviors, making safety a first-class requirement for RL deployment.

Offline RL (Kumar et al., 2020) provides a promising step toward safer deployment, as it allows agents to be pretrained on large curated datasets such as D4RL (Fu et al., 2021a), RL-unplugged (Gulcehre et al., 2020), and DSRL (Liu et al., 2023a). By leveraging prior experience, offline RL agents can achieve high performance without direct online exploration. However, naively deploying a pretrained model is not sufficient: at test time, the environment often differs in unpredictable ways, and the resulting distribution shift may drive the agent into unsafe regions. Recent work has highlighted this challenge by modeling test-time uncertainty and partial observability (Ghosh et al., 2022a, 2021), showing that even for identical observations, transition can vary significantly due to latent dynamics.

To address this, prior approaches have pursued two directions. Belief-based adaptation augments the policy with latent belief variables to model uncertainty, but requires retraining from scratch for effective deployment. Conservative or constrained RL methods introduce penalties to reduce risk, but often sacrifice performance or require fine-tuning with additional cost signals. These limitations make it difficult to align pretrained offline RL agents with safety requirements at test time in a practical and scalable way.

In parallel, the field of large language models (LLMs) has demonstrated that pretrained models can be efficiently adapted to new objectives via *self-alignment*, where models leverage their own reasoning ability to generate instruction-like prompts without human supervision (Sun et al., 2023). Inspired by this idea, we ask: *can offline RL agents be self-aligned for safety at test time, using only their pretrained distribution and without any retraining?*

We propose *Self-Alignment for Safety (SAS)*, a transformer-based framework for safe test-time adaptation without additional training. SAS is based on two key insights: (i) Lyapunov stability can be enforced through occupancy measures, allowing the agent to

Proceedings of the 29th International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300. *Equal contribution. †Corresponding author.

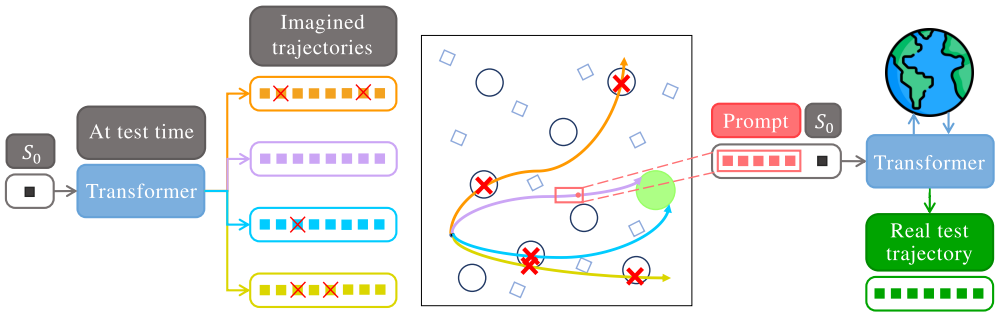

Figure 1: *SAS overview.* From a fixed initial state, the transformer imagines multiple rollouts, flags risky state–action pairs using the Lyapunov condition with (✖), and extracts a safe segment as a prompt to guide the real test-time trajectory (hazards: black ◯, blue ◇; goal: green ●).

identify control-invariant trajectories that avoid unsafe regions; and (ii) transformer-based RL naturally admits a hierarchical RL interpretation, where test-time prompting corresponds to Bayesian inference over latent high-level skills. These enable SAS to generate Lyapunov-guided prompts from imagined trajectories, and align agent behavior safely through in-context learning. Our contributions are summarized as follows.

- A new formulation of Lyapunov stability for offline RL as an occupancy-measure criterion.
- A hierarchical RL interpretation of transformer-based RL via Bayesian inference over latent skills.
- Our method, SAS uses Lyapunov-guided prompts for safe test-time adaptation.
- Empirical evidence that SAS outperforms safe RL baselines, reducing cost and failure by up to two times while maintaining or improving return.

2 PRELIMINARIES

2.1 Problem Setting

We consider a discounted Markov Decision Process (MDP) $\langle \mathcal{S}, \mathcal{A}, r, c, \mathcal{P}, p_1, \gamma \rangle$, where $\mathcal{S}$ and $\mathcal{A}$ are the state and action spaces. Reward and cost are $r, c : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, and the transition kernel $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ gives the next-state distribution. p_1 is the initial state distribution and $\gamma \in (0, 1)$ the discount factor. We assume the per-step cost is uniformly bounded:

$$0 \le c(s, a) \le C_{\max}, \quad \forall (s, a) \in \mathcal{S} \times \mathcal{A}. \tag{1}$$

This assumption is standard in constrained MDP (CMDP) and ensures the discounted cost is well-defined.

We also introduce a latent skill space $\mathcal{Z}$ and consider hierarchical policies. The high-level policy $\pi_\theta^{\text{high}}(z_t \mid s_t, z_{t-1})$ selects a new skill $z_t \in \mathcal{Z}$ given the current state and the previously chosen skill, and the low-level policy $\pi_\phi^{\text{low}}(a_t \mid s_t, z_t)$ executes environment actions conditioned on both the state and the selected skill. For simplicity, we use the notation π to denote the overall policy, which encompasses both the high-level policy π^{high} and the low-level policy π^{low}.

Hierarchical RL as Probabilistic Inference. We interpret hierarchical RL as a probabilistic graphical model (PGM). The joint distribution $p(\tau)$ over a trajectory $\tau = (s_{1:T}, a_{1:T}, z_{1:T})$ factorizes as

$$p_1(s_1) \prod_{t=1}^{T} \mathcal{P}(s_{t+1} \mid s_t, a_t) \underbrace{p(a_t \mid s_t, z_t)}_{\pi_\phi^{\text{low}}} \underbrace{p(z_t \mid s_t, z_{t-1})}_{\pi_\theta^{\text{high}}},$$

where z_t is a latent skill, $p_1(s_1)$ is the initial state distribution, and $\mathcal{P}$ denotes the transition kernel.

We remark that for $t = 1$, the term z_0 is undefined; here we adopt a mild abuse of notation, treating $\pi^{\text{high}}(z_1 \mid s_1, z_0)$ as $\pi^{\text{high}}(z_1 \mid s_1)$ (equivalently, z_0 can be regarded as drawn from a prior). Following the control-as-inference framework (Levine, 2018), we also introduce the *optimality variable* $\mathcal{O}_t \in \{0, 1\}$, a binary observable that encodes reward-based optimality. Later, we will define additional binary observables for Lyapunov or safety conditions. While all such observables are binary indicators, they differ in their role: $\mathcal{O}_t$ connects rewards to the PGM, whereas the additional variables capture safety-related constraints. This unified PGM view, including the observable nodes, is illustrated in Figure 2.

2.2 Occupancy Measure for Safe RL

Transformer-based RL can autoregressively predict actions and, when paired with a generative model such as a VAE (Kingma and Welling, 2013), also generate plausible future states. This allows us to construct imagined trajectories, i.e., rollouts that simulate how

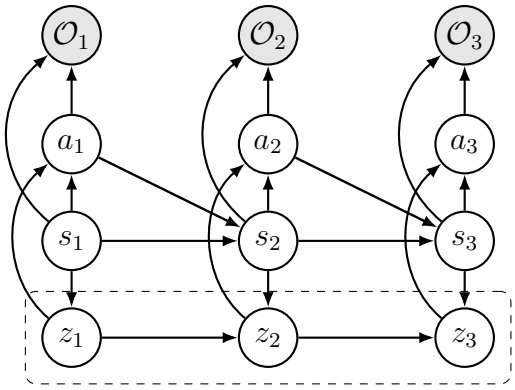

Figure 2: Hierarchical RL as probabilistic inference: a hidden Markov model with latent skills z_t, the optimality variable $\mathcal{O}_t$ linking rewards, and observables for safety conditions.

the agent would behave under the learned dynamics. In our setting, the VAE decoder serves as a *world model* and is integrated into a Decision Transformer (Chen et al., 2021) (see Figure 5).

Given this pretrained transformer with its integrated world model, we define the *model-based occupancy measure* $\hat{\rho}^\pi$, which approximates the true occupancy measure by using imagined trajectories generated from the learned model. Formally, let $\hat{\mathcal{P}}(s' \mid s, a)$ denote the learned transition model (the VAE decoder) and $\pi(a \mid s)$ the policy induced by the transformer, where we write $\pi(a \mid s) = \sum_z \pi^{\text{low}}(a \mid s, z)\, \pi^{\text{high}}(z \mid s, z_{-1})$ and omit z for brevity. Then [1]

$$\hat{\rho}^\pi(s,a) = \sum_{t=1}^{\infty} \gamma^{t-1} \mathop{\mathbb{P}}_{\pi,\hat{\mathcal{P}}}\big(s_t = s,\ a_t = a \mid s_1 \sim p_1\big). \quad (2)$$

We use $\mathcal{P}$ for the true environment and $\hat{\mathcal{P}}$ for the learned world model dynamics.

Connection to Safe RL. We formulate safe RL as a constrained optimization problem:

$$\max_\pi J_R(\pi) \quad \text{s.t.} \quad J_C(\pi) \le d, \quad (3)$$

where $J_R(\pi) = \mathbb{E}_{(s,a)\sim\rho^\pi}[\, r(s,a)\,]$ and $J_C(\pi) = \mathbb{E}_{(s,a)\sim\rho^\pi}[\, c(s,a)\,]$. This form shows that the optimal policy π^* must assign low occupancy to high-cost regions; in other words, safety constraints act directly on the distribution of visited state–action pairs.

Generalization with Offline Data. Offline datasets are often heterogeneous, containing both expert and medium-quality trajectories. Following Rashidinejad et al. (2021), we introduce the best concentrability coefficient $D_{\text{conc}} \ge 1$, the smallest constant:

$$\hat{\rho}^{\pi^*}(s,a) \ \le\ D_{\text{conc}}\, \rho_{\text{data}}(s,a), \quad \forall (s,a) \in \mathcal{S}\times\mathcal{A}$$

and $\rho_{\text{data}} > 0$. A smaller D_{conc} indicates that the dataset provides good coverage and thus tighter safety guarantees, whereas a larger D_{conc} reflects sparse or biased data and results in weaker guarantees. Combining this bound with the cost boundedness in Equation (1), we obtain the conservative sufficient condition

$$\mathcal{R}_\rho \ =\ \Big\{(s,a)\ \Big|\ \hat{\rho}^\pi(s,a) \ \ge\ \tfrac{d}{C_{\max} D_{\text{conc}}}\Big\}, \quad (4)$$

which yields a tractable feasible region for satisfying the CMDP constraint in Equation (3).

Interpretation. Equation (3) reveals an inverse relationship between occupancy and cost: a safe policy must avoid placing significant probability mass in high-cost regions. D_{conc} adapts this principle to the offline setting by accounting for dataset coverage.

2.3 Lyapunov Density and Control-Invariant Sets for Safety

Building on the feasible set $\mathcal{R}_\rho$, we now turn to the **Lyapunov Density Model (LDM)** (Kang et al., 2022), which formalizes safety as density-based monotonicity and provides a novel way to define control-invariant sets. Practically, restricting the agent to actions within $\mathcal{R}_\rho$ ensures that the safety constraint in Equation (3) is satisfied, and Figure 1 illustrates how imagined trajectories are filtered using both occupancy measures and Lyapunov conditions. Unlike classical state-only control Lyapunov functions (CLFs), LDM is formulated in a (s,a)-based manner: the Lyapunov surrogate $G(s,a)$ is **learned from offline data**. LDM posits a Lyapunov-like function $G : \mathcal{S}\times\mathcal{A} \to \mathbb{R}_{\ge 0}$ satisfying:

1. $G(s_e, a_e) = 0$ for an equilibrium (s_e, a_e)
2. $G(s,a) > 0$ for all $(s,a) \ne (s_e, a_e)$
3. $G(s,a) \ \ge\ \gamma \mathbb{E}_{s'\sim\mathcal{P}(\cdot|s,a)}\big[\min_{a'} G(s',a')\big]$.

These conditions ensure that G is *non-increasing* along feasible trajectories: whenever (s',a') is the next state–action pair reached from (s,a) by the following *greedy policy*, the value of G cannot increase. **LDM backup operator** is defined as a Bellman-style recursion adapted to enforce Lyapunov density conditions:

$$\big(\mathcal{T}_{\text{LDM}} G\big)(s,a) = \max\Big\{ -\log\hat{\rho}^\pi(s,a),\ \gamma\, \mathbb{E}_{s'\sim P(\cdot|s,a)}\big[\min_{a'} G(s',a')\big]\Big\}$$

[1] Unnormalized version. The $(1-\gamma)$ factor is omitted since it can be canceled in CMDP objectives.

where the first term ties G to the estimated density, and the second term enforces discounted Lyapunov descent under the learned dynamics. The max selects the more conservative of the two constraints.

Control-Invariant Set for Safety. Define the control (forward)-invariant set as

$$\mathcal{R}_c = \{(s,a) \mid G(s,a) \le c\}, \quad c > 0. \tag{5}$$

G is non-increasing along feasible trajectories. Thus, a *greedy policy* such as $\arg\min_a G(s,a)$ cannot escape from $\mathcal{R}_c$, and every trajectory remains inside while asymptotically converging to the equilibrium point.

3 FROM OFFLINE RL TO LYAPUNOV-BASED SAFETY EVALUATION

In this section, we establish a principled transition from offline RL to Lyapunov-based safety evaluation. While offline RL provides a pretrained policy distribution, it does not guarantee safety at deployment, as trajectories may still enter unsafe regions due to limited coverage. To address this, we reinterpret Lyapunov stability in terms of occupancy measures estimated from offline RL models, and introduce our Lyapunov model G_{SAS} as a principled way to evaluate and enforce safety. This section formalizes the connection between safe RL and occupancy measures, presents our core Lyapunov model, and develops a probabilistic inference framework that enables trajectory-level search for safety.

3.1 Lyapunov Model for Offline Safe RL

Our central contribution in this section is to extend the Lyapunov Density Model (LDM) with occupancy measures, yielding a Lyapunov model tailored for offline safe RL. We define the energy function based on the estimated occupancy measure $\hat{\rho}^\pi$ in Equation (2):

$$E(s,a) = -\log \hat{\rho}^\pi(s,a),$$

which serves as an energy function that provides an upper bound for the control-invariant set. Controlling with the learned LDM guarantees that the agent stays within this set. Extending this idea, we define our Lyapunov model for safety evaluation as

$$\begin{aligned} G_{\text{SAS}}(s,a) &= \min_{\pi} \max_{t'} \Big[E(s_{t'}, \pi(s_{t'})) - E(s,a) \Big] \\ &= \log \hat{\rho}^\pi(s,a) - \underbrace{\max_{\pi \in \Pi} \min_{t' \in \{1,\dots,T\}} \log \hat{\rho}^\pi\big(s_{t'}, \pi(s_{t'})\big)}_{\text{related to } R_\rho \text{ in Eq. (4)}}. \end{aligned} \tag{6}$$

By construction, G_{SAS} maximizes the lowest occupancy encountered along any trajectory, thereby preventing the policy from collapsing into unsafe, low-density regions. This induces a control-invariant set:

$$\mathcal{R}_G^{\text{SAS}} = \big\{(s_t,a_t) \,\big|\, G_{\text{SAS}}(s_t,a_t) \le -\log \tfrac{d}{C_{\max} D_{\text{conc}}}\big\}.$$

Remark. G_{SAS} refines the offline RL model by enforcing Lyapunov stability in terms of data-supported occupancies. This guarantees that imagined trajectories remain within a safe invariant set without retraining or fine-tuning. Two key implications follow from these properties: **(1) Conservative behavior.** Since G_{SAS} is non-increasing along feasible trajectories, policies are naturally restricted to high-density regions of the offline data, reducing the risk of out-of-distribution actions. **(2) Test-time self-alignment.** In our framework, densities and occupancy measures are estimated from offline data and imagined rollouts (via transformer-generated trajectories), so G_{SAS} serves as a prior that guides prompt selection and action choices toward the safe region $\mathcal{R}_G^{\text{SAS}}$. We will elaborate on how this prior enables test-time self-alignment in Section 4.

3.2 Lyapunov Condition as Probabilistic Inference

Additional training on top of the pretrained model is computationally inefficient. Instead, we reformulate our Lyapunov model for safety evaluation, $G_{\text{SAS}}(s,a)$ in Equation (6), as a probabilistic inference that can be integrated into the transformer's in-context learning.

By introducing observable indicators of the Lyapunov condition, safety verification can then be cast as maximizing their log-likelihood. This reformulation provides a principled criterion to rank and select safe *imagined* trajectories without any additional retraining. To formalize this reformulation, we first define observable indicators that encode whether the Lyapunov condition is satisfied along a trajectory.

Definition 1 (Lyapunov-Condition Observables)**.** To make the Lyapunov condition verifiable, we introduce two indicator observables:

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

Using these observables, our Lyapunov model in Equation (6) can be shown to be equivalent to maximizing their log-likelihood, as stated in the following theorem.

Theorem 2. *With the observables in Definition 1, the Lyapunov formulation in Equation (6) is (up to an additive constant) equivalent to*

$$\max_{\tau} \frac{1}{T} \sum_{t=1}^{T} \big[\log P(\mathcal{U}_t{=}1 \mid \tau) + \log P(\mathcal{V}_t{=}1 \mid \mathcal{U}_t{=}1, \tau) \big].$$

Here, T denotes the trajectory horizon and τ an imagined trajectory generated by the pretrained model. The full proof is given in Appendix G.

This theorem establishes that the Lyapunov condition can be reformulated as a probabilistic inference objective, which allows safety to be assessed and enforced directly on imagined trajectories. To compute G_{SAS} while simultaneously selecting an optimal trajectory, we relax the problem into the iterative imagination-based procedure described in Algorithm 1 of Section 4.

4 LYAPUNOV-GUIDED SELF-ALIGNMENT

In Section 3, we introduced the Lyapunov condition as a criterion for control-invariant sets in offline RL. Here, we ask: how can this condition be used to realign a pretrained agent *without retraining*? Our key idea is to view Lyapunov evaluation as an inference problem over imagined trajectories, so that a pretrained model can be guided toward safer behavior via self-generated prompts. We consider two complementary axes: **(1) Occupancy-based evaluation:** filtering trajectories with low estimated density, thereby mitigating error accumulation along the *value horizon*. **(2) Trajectory-based prompting:** Consider the decay of G_{SAS} as a binary observable, and recycling safe segments as in-context prompts to shorten the *policy horizon*. Together, these axes constrain imagined rollouts and enable prompt-based alignment.

4.1 Self-alignment via in-context learning

Our procedure, **SAS (Self-Alignment for Safety)**, integrates Lyapunov filtering with transformer imagination through three steps: (1) evaluate imagined rollouts with $G_{\mathrm{SAS}}(s,a)$, (2) select safe fragments within $\mathcal{R}_G^{\mathrm{SAS}}$, (3) insert them back as prompts. This jointly reduces value and policy horizons and enables safe test-time adaptation *without parameter updates*. We next formalize the inference view underlying SAS.

When aligning a transformer-based model to a downstream task, the model adapts by conditioning on a prompt composed of demonstration examples. This capability, known as *in-context learning*, can be interpreted as implicit Bayesian inference over latent concepts (Xie et al., 2021). Formally, the posterior predictive distribution $p(\texttt{output} \mid \texttt{prompt})$ is

$$\int p(\texttt{output} \mid \texttt{prompt}, \theta)\, p(\theta \mid \texttt{prompt})\, d\theta,$$

where θ denotes a latent concept that parameterizes the hidden Markov model $p(\texttt{output} \mid \texttt{prompt}, \theta)$. Intuitively, conditioning on a prompt selects a particular θ, so that $p(\texttt{output} \mid \texttt{prompt}, \theta)$ generates aligned behavior. In our framework, we draw an analogy between this latent concept θ and the parameter of a high-level policy in hierarchical RL, with the low-level policy handling primitive actions. This perspective motivates the probabilistic formulation of SAS in the following.

4.2 In-context Bayesian Inference for Prompt-based Policy Extraction

We now show how the policy that generates prompts in our self-alignment framework can be identified within the transformer through an in-context Bayesian inference view. This enables us to connect prompt-based self-alignment with Bayesian inference and to derive probabilistic guarantees on the resulting trajectories.

Here, unlike in the LLM setting where θ denotes a latent concept, we interpret θ as the parameter of the high-level policy $\pi_\theta^{\mathrm{high}}$ in our hierarchical RL formulation. At test time, the first latent skill $\mathbf{z}_1^{\mathrm{test}}$ is sampled along with a trajectory τ starting from the initial state $\mathbf{s}_1^{\mathrm{test}}$, conditioned on a prompt $\mathbf{p}_{1:K}$, which is a K-length demonstration prefix of state–action pairs $(\mathbf{s}_{-K+1}, \mathbf{a}_{-K+1}, \ldots, \mathbf{s}_0, \mathbf{a}_0)$ (see Figure 2). Intuitively, such a prompt selects a feasible high-level policy $\pi_\theta^{\mathrm{high}}$ from the space implicitly encoded in the offline dataset $\mathcal{D}$. Our goal is to recover the safe high-level policy $\pi_{\theta^*}^{\mathrm{high}}$ consistent with the demonstration prompt. To formalize this inference, we introduce the Lyapunov observables $\mathcal{U}_t$ and $\mathcal{V}_t$ from Section 3.2 and define the combined optimality variable

$$\mathcal{O}_t := \mathcal{U}_t \wedge \mathcal{V}_t, \tag{7}$$

so that $\mathcal{O}_t = 1$ indicates both Lyapunov conditions hold at time t. The inference distribution over trajectories that satisfy these conditions is For brevity, we drop the superscript $^{\mathrm{test}}$ on test-time variables in the following.

$$\begin{aligned} &p(\mathcal{O}_{\mathrm{traj}} = 1 \mid \mathbf{p}_{1:K}, \mathbf{s}_1) \\ &= \int_\theta \sum_{z_1 \in \mathcal{Z}} g_{\pi_\theta}(\tau \mid z_1, s_1)\, p(z_1 \mid \mathbf{p}_{1:K}, s_1, \theta) \\ &\quad \times \prod_{t=1}^{T} p(\mathcal{O}_t \mid s_t, a_t)\, e^{K r_K(\theta)}\, p(\theta)\, d\theta. \end{aligned} \tag{8}$$

where the rollout likelihood under π_θ is defined as

$$\begin{aligned} g_{\pi_\theta}(\tau \mid z_1^{\mathrm{test}}, s_1^{\mathrm{test}}) = \prod_{t=1}^{T} \Big[&\hat{\mathcal{P}}(s_{t+1}^{\mathrm{test}} \mid s_t^{\mathrm{test}}, a_t^{\mathrm{test}}) \\ &\times \pi_\phi^{\mathrm{low}}(a_t^{\mathrm{test}} \mid s_t^{\mathrm{test}}, z_t^{\mathrm{test}})\, \pi_\theta^{\mathrm{high}}(z_t^{\mathrm{test}} \mid s_t^{\mathrm{test}}, z_{t-1}^{\mathrm{test}}) \Big] \end{aligned}$$

(with s_1^{test} suppressed in subsequent uses for brevity), and

$$r_K(\theta) = \tfrac{1}{K} \log \frac{p(\mathbf{p}_{1:K},\ s_1^{\mathrm{test}} \mid \theta)}{p(\mathbf{p}_{1:K},\ s_1^{\mathrm{test}} \mid \theta^*)} \tag{9}$$

measures the compatibility between the prompt $\mathbf{p}_{1:K}$ and θ. Here, θ^* is for the high-level policy from which the prompt $\mathbf{p}_{1:K}$ was generated, serving as the safe reference policy in the denominator of $r_K(\theta)$. A trajectory that satisfies Theorem 2 is equivalent to the one induced by the safe reference policy $\pi^{\text{high}}_{\theta^*}$, which maximizes the likelihood of $\mathcal{O}_{\text{traj}} = 1$.

This formulation extends the implicit Bayesian inference view (Xie et al., 2021) in two ways: (i) it explicitly incorporates the low-level policy π^{low}_{ϕ} to model the action space in RL, and (ii) it leverages the predictive transition model $\hat{\mathcal{P}}(s_{t+1} \mid s_t, a_t)$ to generate imagined trajectories consistent with the environment. To connect the inference formulation with Lyapunov-based safety, we now establish a probabilistic guarantee for the trajectory returned by Algorithm 1.

Assumption 3. The Lyapunov difference $\|G_{\text{SAS}}(s_t, a_t) - G_{\text{SAS}}(s_{t+1}, a_{t+1})\|$ is bounded by a constant $L > 0$ for all transitions.

Theorem 4. *Under Assumption 3, for τ_{best} returned by Algorithm 1, the probability of leaving $\mathcal{R}^{\text{SAS}}_G$ is bounded by*

$$P\big[\tau_{\text{best}} \not\subset \mathcal{R}^{\text{SAS}}_G\big] \;\le\; \left(\frac{\mathbb{E}_{(s,a)\sim\mathcal{D}}\big[-\log \hat{\rho}(s,a)\big]}{c_2}\right)^{NT} + \exp\!\left(-\frac{2M\kappa^2(c_2-c_1)^2}{TL^2}\right). \tag{10}$$

The proof is in Appendix G.4. Here, $c_1, c_2 > 0$ are constants defining the control-invariant set and κ is a scaling constant from Hoeffding's inequality. As the iterations $N, M \to \infty$, the RHS of Equation (10) vanishes, implying that τ_{best} remains in $\mathcal{R}^{\text{SAS}}_G$ with high probability.

Practical interpretation. The first term decays exponentially in $N \times T$: increasing the number of imagined rollouts (N) or the trajectory length (T) tightens the energy-based feasibility check (condition $\mathcal{U}_t$). The second term decays exponentially in M: more re-imagination trials strengthen the Lyapunov-descent check (condition $\mathcal{V}_t$). Together, the bound confirms that SAS is not merely a heuristic but admits a formal probabilistic safety guarantee whose tightness improves with moderate computational budget (see ablations in Appendix E).

4.3 Safe-Alignment for Safety (SAS)

Algorithm 1 shows our Lyapunov-guided self-alignment procedure. First, from a fixed test initial state, we construct an initial prompt $\hat{\mathbf{p}}_{1:K}$ using a conservative min–max rule on the energy $E(s,a) = -\log \hat{\rho}^{\pi}(s,a)$ for condition $\mathcal{U}_t$. Second, conditioned on this prompt, we re-imagine trajectories and select the one maximizing the cumulation $\sum_t \mathcal{V}_t$. The resulting $\mathbf{p}_{1:K}$, combined with the initial state, defines safe test-time behavior.

Algorithm 1 **S**elf **A**lignment for **S**afety (SAS): Prompt generation for safe test-time adaptation

Require: Pretrained transformer from $\mathcal{D}$; test initial state $\mathbf{s}_1^{\text{test}}$; prompt length K

1: **for** $i = 1, 2, \ldots, N$ **do** ▷ loop for condition $\mathcal{U}_t$
2: Imagine trajectory $\tau^{(i)} \sim p(\tau \mid \mathbf{s}_1^{\text{test}})$
3: Compute E_t for all t in $\tau^{(i)}$
4: $\hat{E}_i,\ t_i \leftarrow \max_t E_t,\ \arg\max_t E_t$
5: **end for**
6: $i^* \leftarrow \arg\min_i \hat{E}_i$
7: $\hat{\mathbf{p}}_{1:K} \leftarrow \tau^{(i^*)}[t_{i^*}-K+1 : t_{i^*}]$

8: **for** $j = 1, 2, \ldots, M$ **do** ▷ loop for condition $\mathcal{V}_t$
9: Imagine trajectory $\tau^{(j)} \sim p(\tau \mid \hat{\mathbf{p}}_{1:K}, \mathbf{s}_1^{\text{test}})$
10: Compute E_t for all t in $\tau^{(j)}$
11: $t_j \leftarrow \arg\max_t E_t$
12: $v_j \leftarrow \sum_t \mathcal{V}_t$
13: **end for**
14: $j^* \leftarrow \arg\max_j v_j$
15: $\mathbf{p}_{1:K} \leftarrow \tau^{(j^*)}[t_{j^*}-K+1 : t_{j^*}]$
16: **return** $\mathbf{p}_{1:K}$ and deploy $\pi^{\text{high}}_{\theta^*}$ by conditioning on $(\mathbf{p}_{1:K}, \mathbf{s}_1^{\text{test}})$

In Algorithm 1, lines 1–6 choose $i^* = \arg\min_i \max_t E_t$ among rollouts from $\mathbf{s}_1^{\text{test}}$ and set the initial prompt as the last K pairs at t_{i^*} (line 7). Lines 9–13 then evaluate $v_j = \sum_t \mathcal{V}_t$ for each conditioned rollout. Lines 14–15 pick $j^* = \arg\max_j v_j$ and form the final prompt from the K pairs at t_{j^*}. Finally, line 16 conditions the transformer on $(\mathbf{p}_{1:K}, \mathbf{s}_1^{\text{test}})$ to yield the safe policy.

5 EXPERIMENTS

In this section, we empirically evaluate the performance of SAS, comparing it to vanilla offline RL, prior offline safe RL, and other test-time adaptation methods in the `MuJoCo` (Brockman et al., 2016), `Safety Gymnasium` (Ji et al., 2023), and `Bullet-Safety-Gym` (Gronauer, 2022) benchmarks. We also present analyses and ablation on the policy extraction under self-alignment.

5.1 Experimental Setup

We employ the D4RL dataset (Fu et al., 2021b) for `MuJoCo` and the DSRL dataset (Liu et al., 2023a) for `Safety Gymnasium`. Both reward and cost returns are normalized for comparability across tasks. The methods denoted by *SAS* correspond to our self-alignment method. Throughout, *'+SAS'* denotes our Lyapunov-guided self-alignment applied to a pretrained back-

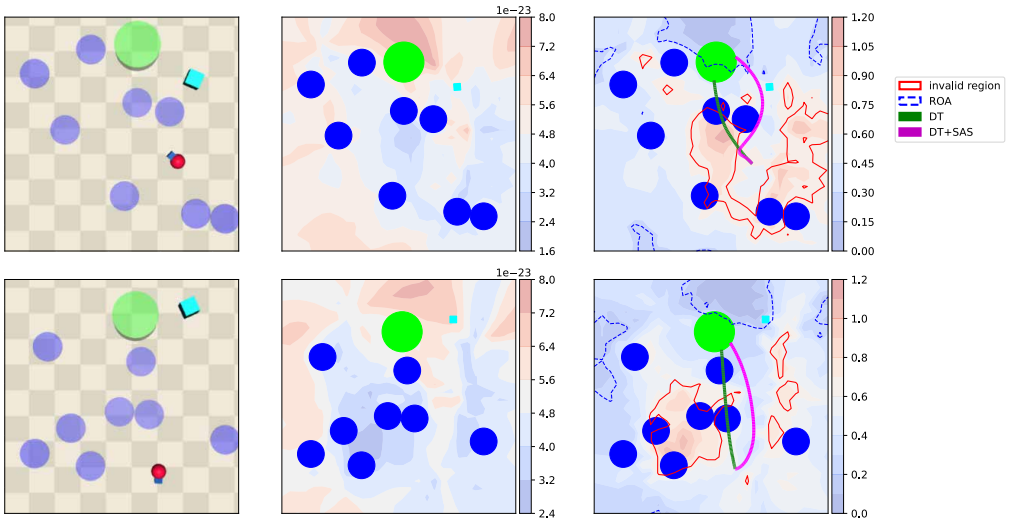

Figure 3: **SAS dodges hazard better. Left:** env. with fixed **hazards**, a movable **obstacle**, and a **goal**. **Middle:** The state density from action-averaged occupancy measure. **Right:** Trajectories with and without SAS, overlaid on G_{SAS} landscape. Blue regions denote control-invariant set $\mathcal{R}_G^{\text{SAS}}$, red lines indicate unsafe areas.

bone (DT or CDT) that includes our VAE-augmented world model for imagination. We do not report a standalone *'SAS'* without a backbone; all our results are backbone+SAS. For brevity, task names are abbreviated, e.g., `PointGoal1` (PG1), `PointPush1` (PP1), and `CarButton2` (CB2). Detailed experimental settings are provided in Appendix C due to space limitations.

5.2 Results and Discussions

We present our experimental results and corresponding insights in a Q&A style. Although referenced in the discussion for clarification, figures and tables not included in the main text are provided in Appendix F.

Q1: Why is SAS safer at test time?

Figure 3 illustrates how SAS guides safer behavior in the `PointGoal1-v0` task from `Safety Gymnasium` benchmark (Ji et al., 2023). As shown in the left column, the agent (●) is initialized at a random position with a random orientation. In the right column, the resulting trajectories illustrate that DT+SAS, unlike DT, reaches the goal (●) without entering any hazard (●), thus ensuring a safe trajectory toward the target.

We define the *action-averaged occupancy measure* as the average of $\hat{\rho}^\pi(s,a)$ over actions $\pm x, \pm y$ at each state, since actions represent the intended movement direction of the agent. Similarly, $G_{\text{SAS}}(\mathbf{s}_t, \mathbf{a}_t)$ is averaged in the same manner. The *invalid region* corresponds to the area where G_{SAS} exceeds its 95th percentile, while the *region of attraction (ROA)* denotes a tight control-invariant set defined by the 10th percentile of G_{SAS}, which tends to concentrate near the goal (●).

The safer behavior of DT+SAS can be explained by comparing the action-averaged occupancy measure in the middle column with G_{SAS} in the right column. While G_{SAS} assigns high values to directions leading toward hazard-dense areas (●), thereby forming invalid regions, the action-averaged occupancy measure yields more blurry boundaries. Consequently, DT, having limited foresight, often enters such regions, ultimately incurring costs by stepping into hazards.

Although neither DT nor DT+SAS explicitly uses cost, both are trained on expert data from the offline safe RL dataset DSRL, where agents learn to avoid hazards and reach the goal. However, the occupancy measure $\hat{\rho}^\pi$ of DT exhibits myopic behavior in long-horizon evaluation, deeming current actions as probable without sufficient consideration of future risks. By contrast, Lyapunov-guided self-alignment in SAS refines the policy at test time, enabling deployment of an agent that is both safer and more aligned with the offline distribution.

Q2: Does SAS consistently improve?

Table 2: Performance of DT and DT+SAS in `MuJoCo`.

Environment		Hopper		Walker2d		Humanoid	
		expert	medium	expert	medium	expert	medium
DT	reward	110.7	86.6	107.7	82.2	98.5	40.5
	failure	0.05	1	0	0.54	0.20	0.97
DT+SAS	reward	110.7	**87.5**	107.7	**89.5**	**103.5**	**50.6**
	failure	**0.03**	1	0	**0.46**	**0.10**	**0.87**

First, Table 2 and Table 9, whose rewards are normalized following (Kumar et al., 2020), demonstrate that SAS yields notable improvements in both reward (return) and failure rate (safety) even in general offline RL without explicit cost annotations, such as D4RL.

Table 1: Full results in `Safety Gymnasium`. Values are averaged over three cost thresholds, 20 evaluation episodes, and three random seeds. Gray: unsafe agents. **Bold**: safe agents with normalized cost less than 1. **Blue**: safe agents achieving the highest reward.

Task	DT + SAS		CDT + SAS		CDT		BC-All		BC-Safe		BCQ-Lag		BEAR-Lag		CPQ		COptiDICE		DCRL	
	reward	cost	reward	cost	reward	cost	reward	cost	reward	cost	reward	cost	reward	cost	reward	cost	reward	cost	reward	cost
PointGoal1	0.66	1.19	0.65	1.27	0.69	1.12	**0.65**	**0.95**	**0.43**	**0.54**	**0.71**	**0.98**	0.74	1.18	**0.57**	**0.35**	0.49	1.66	**0.24**	**0.86**
PointGoal2	0.65	1.78	**0.52**	**0.94**	0.59	1.34	0.54	1.97	**0.29**	**0.78**	0.67	3.18	0.67	3.11	0.4	1.31	0.38	1.92	**0.28**	**0.26**
PointPush1	**0.28**	**0.62**	**0.26**	**0.54**	**0.24**	**0.48**	**0.19**	**0.61**	**0.13**	**0.43**	**0.33**	**0.86**	**0.22**	**0.79**	**0.2**	**0.83**	**0.13**	**0.83**	**0.01**	**0.52**
PointPush2	**0.24**	**0.64**	**0.20**	**0.53**	**0.21**	**0.65**	**0.18**	**0.91**	**0.11**	**0.8**	**0.23**	**0.99**	0.16	0.89	0.11	1.04	0.02	1.18	**0.02**	**0.07**
PointButton1	0.49	1.38	0.51	1.27	0.5	1.68	0.1	10.5	**0.06**	**0.52**	0.24	1.73	0.2	1.6	0.69	3.2	0.13	1.4	**0.01**	**0.48**
PointButton2	0.51	1.14	**0.41**	**0.98**	0.46	1.57	0.27	2.02	0.16	1.1	0.4	2.66	0.43	2.47	0.58	4.3	0.15	1.51	**0.18**	**0.64**
CarGoal1	**0.67**	**0.85**	**0.65**	**0.90**	0.66	1.21	**0.39**	**0.33**	**0.24**	**0.28**	**0.47**	**0.78**	0.61	1.13	0.79	1.42	**0.35**	**0.54**	**0.35**	**0.88**
CarGoal2	0.48	1.15	**0.42**	**0.98**	0.48	1.25	0.23	1.05	**0.14**	**0.51**	0.3	1.44	0.28	1.01	0.65	3.75	**0.25**	**0.91**	0.11	2.51
CarPush1	**0.31**	**0.51**	**0.31**	**0.49**	**0.31**	**0.4**	**0.22**	**0.36**	**0.14**	**0.33**	**0.23**	**0.43**	**0.21**	**0.54**	**-0.03**	**0.95**	**0.23**	**0.5**	**-0.1**	**0.09**
CarPush2	0.22	1.16	**0.21**	**0.75**	0.19	1.3	**0.14**	**0.9**	**0.05**	**0.45**	0.15	1.38	0.1	1.2	0.24	4.25	0.09	1.07	**-0.13**	**0.17**
CarButton1	0.17	1.08	**0.27**	**0.98**	0.21	1.6	0.03	1.38	**0.07**	**0.85**	0.04	1.63	0.18	2.72	0.42	9.66	-0.08	1.68	**0.12**	**0.95**
CarButton2	**0.14**	**0.84**	0.30	1.11	0.13	1.58	-0.13	1.24	**-0.01**	**0.63**	0.06	2.13	-0.01	2.29	0.37	12.51	-0.07	1.59	0.09	1.42
Average	0.402	1.03	0.392	**0.895**	0.389	1.18	0.234	1.85	0.151	**0.602**	0.319	1.52	0.316	1.58	0.416	3.63	0.172	1.23	0.098	**0.737**

Here, the failure rate is defined as the *early termination* event in `MuJoCo`, where an episode ends prematurely due to exceeding internal unhealthy cost thresholds before reaching the maximum episode length.

As shown in Table 8, SAS achieves substantially lower costs in `Safety Gymnasium` during test-time adaptation. Note that DT does not leverage cost information in its offline pretraining dataset, while CDT incorporates cost signals as in the original pretraining. Surprisingly, in most environments, both backbones become safer once our density-based Lyapunov guidance is applied. This observation is analogous to the results of LDM, yet our method attains safety improvements without requiring a new from-scratch pretraining, offering a simple and effective enhancement.

Finally, Table 1 and Table 7 report aggregated benchmark results in safe RL. Table 7 extends Table 1 by including 8 additional tasks from `bullet-safety-gym` on top of the 16 tasks in `Safety Gymnasium`. Across heterogeneous agents with varying observations and dynamics (`point, car, ant, ball, drone`), SAS consistently outperforms, including strong safe-control methods such as BC-Safe, and yields superior results over its own backbones DT and CDT. Among the methods that satisfy the safety criterion (average normalized cost ≤ 1), namely CDT+SAS, BC-Safe, and DCRL, CDT+SAS achieves the highest average reward. Per-task standard deviations across seeds and cost thresholds are reported in Table 8; SAS consistently reduces variance in cost relative to its backbone.

This effectiveness without explicit cost signals can be attributed to the structure of the offline safe RL datasets (e.g., DSRL), where expert demonstrations naturally concentrate in safe regions. Consequently, high-density states under the occupancy measure $\hat{\rho}^{\pi}$ largely coincide with low-cost states, allowing the density-based Lyapunov model to serve as an implicit safety proxy.

Q3: Is SAS better than random or max prompt?

In Table 6, DT+SAS exhibits consistently lower costs than the random prompting baseline (DT+rand), whose costs often exceed those of DT in half of `Safety Gymnasium`. To validate the role of Lyapunov-guided selection, we also consider an alternative ablation (DT+maxmax) that chooses the trajectory with the largest maximum step-wise energy. This variant yields higher costs and failure rates, and generally lower rewards, compared to DT+SAS. These results confirm that SAS provides a valid and effective self-generated prompt: by selecting trajectories that minimize the worst-case energy, the transformer makes safer decisions throughout the episode, analogous to choosing an *appropriate high-level skill in hierarchical RL*.

Q4: Does SAS benefit undertrained models?
The *undertrained DT* panel of Figure 4 compares an under-pretrained DT backbone with (*align*) and without (*no align*) self-alignment. SAS improves the reward across all three environments, even though cost and failure rate also rise relative to the well-trained baseline. It implies that SAS can extract useful behavior from partially trained models at test time – the Lyapunov-guided prompt steers the agent toward in-distribution.

Q5: How essential is $\mathcal{V}_t$?
The *without* $\mathcal{V}_t$ panel of Figure 4 ablates the Lyapunov-descent stage by comparing $\mathcal{U}_t$ alone (U) with the full $\mathcal{U}_t \wedge \mathcal{V}_t$ (UV). Adding $\mathcal{V}_t$ consistently lowers cost across all three environments; in terms of failure rate, the improvement carries over to `PointGoal1` and `PointPush2`, with only a slight uptick on `CarGoal1`. This confirms that $\mathcal{V}_t$ is the component stabilizing long-horizon rollouts – structurally, it realizes the second term of Equation (10) governed by M and κ, so removing it (κ=0) tolerates a violation at every step and inflates cost.

Q6: Is SAS robust across design choices?
The remaining five panels of Figure 4 sweep the main

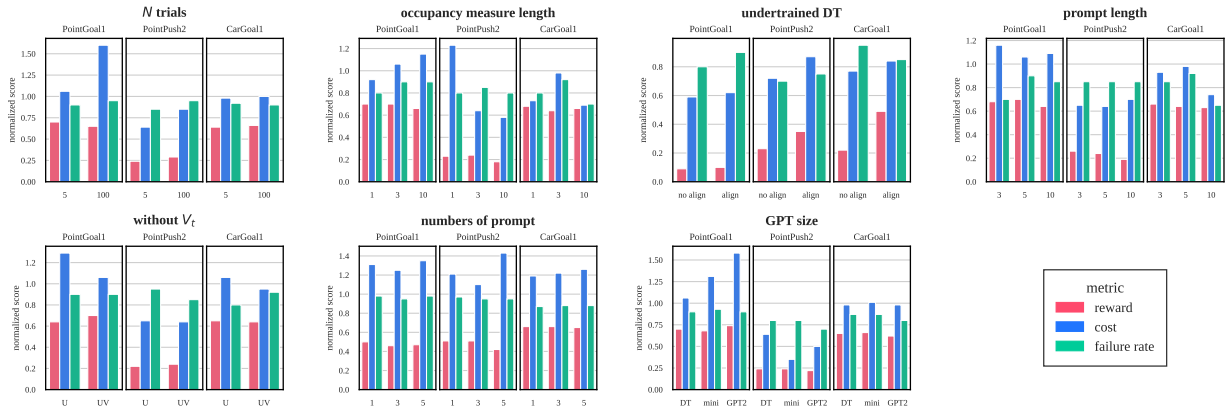

Figure 4: **Comprehensive ablation of SAS.** Normalized reward, cost, and failure rate on `PointGoal1`/`PointPush2`/`CarGoal1`. **Top:** number of imagined trajectories per loop ($N \in \{5, 100\}$); horizon used to aggregate energy $E = -\log \hat{\rho}$ ($\{1, 3, 10\}$); self-alignment on an under-pretrained DT (*no align* vs. *align*); prompt length $K \in \{3, 5, 10\}$. **Bottom:** ablation of the Lyapunov-descent condition ($\mathcal{U}_t$-only [U] vs. $\mathcal{U}_t \wedge \mathcal{V}_t$ [UV]); number of in-context prompt fragments concatenated ($\{1, 3, 5\}$); backbone scale (DT, GPT-mini, GPT-2).

hyperparameters of SAS: number of imagined trajectories $N \in \{5, 100\}$, the energy aggregation horizon $\{1, 3, 10\}$ (*occupancy measure length*), prompt length $K \in \{3, 5, 10\}$, the number of prompt fragments concatenated $\{1, 3, 5\}$ (*numbers of prompt*), and backbone scale (DT $\rightarrow$ GPT-mini $\rightarrow$ GPT-2). Variations within each axis are small and environment-specific, confirming that SAS is broadly robust to these choices. Two of these axes have a direct interpretation through Theorem 4: N tightens the first ($\mathcal{U}_t$) term of the bound, although excessively large N (e.g., 100) can overfit to extremely high-occupancy regions and erode task return; the energy aggregation horizon corresponds to T, where G_{SAS} captures the worst-case energy along a trajectory, so a moderate horizon already pinpoints the decisive unsafe region. The remaining axes (prompt length, number of fragments, backbone size) are not directly tied to the bound but show consistent saturation, in line with the qualitative robustness predicted by the analysis. Per-axis details are deferred to Appendix E.

Related work. SAS sits at the intersection of three lines. *(i) Transformer-based RL*—Decision Transformer (Chen et al., 2021; Janner et al., 2021), transformer world models (Robine et al., 2023; Micheli et al., 2023), and prompt-/cost-conditioning extensions (Jiang et al., 2023; Xu et al., 2022b; Liu et al., 2023b)—provides the autoregressive imagination backbone we adapt, but does not address safety alignment at test time. *(ii) Safe RL with Lyapunov-style certificates* (Chow et al., 2018; Chang et al., 2019; Kang et al., 2022; Qin et al., 2021; Zheng et al., 2024) acts at *training time*, learning a constrained or feasibility-aware policy from scratch. *(iii) Alignment of pretrained models* is dominated by RLHF (Ouyang et al., 2022) and prompt-based instruction tuning (Askell et al., 2021; Sun et al., 2023), but safety alignment of an offline RL agent *at test time* remains largely unexplored. SAS fills this gap by aligning an offline-pretrained transformer at deployment through Lyapunov-guided self-generated prompts, without parameter updates or human supervision. A full discussion is deferred to Appendix A in the appendix.

6 CONCLUSION

We presented SAS, a Lyapunov-guided test-time adaptation for offline RL that improves safety without retraining, consistently lowering cost and failure while maintaining competitive returns. Current limitations include the additional inference cost from multiple imagined rollouts and the reliance on data density rather than explicit cost constraints, making safety guarantees dependent on offline dataset coverage. We hope SAS encourages future work on reducing inference overhead, incorporating explicit constraint signals, and broader test-time adaptation for safety.

Acknowledgments

This work was supported in part by the National Research Foundation of Korea (NRF) (grant numbers RS-2024-00451435 (20%) and RS-2024-00413957 (20%)); the Institute of Information & Communications Technology Planning & Evaluation (IITP) (grant numbers RS-2025-02305453 (15%), RS-2025-02273157 (15%), RS-2025-25442149 (15%), and RS-2021-II211343 (15%)) funded by the Ministry of Science and ICT (MSIT); the Institute of New Media and Communications (INMAC); the BK21 FOUR Program of the Education and Artificial Intelligence Graduate School

Program at Seoul National University; and the Research Program for Future ICT Pioneers at Seoul National University (2026).

References

Askell, A., Bai, Y., Chen, A., Drain, D., Ganguli, D., Henighan, T., Jones, A., Joseph, N., Mann, B., DasSarma, N., et al. (2021). A general language assistant as a laboratory for alignment. *arXiv preprint arXiv:2112.00861*.

Bansal, S. and Tomlin, C. J. (2021). Deepreach: A deep learning approach to high-dimensional reachability. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1817–1824. IEEE.

Bharadhwaj, H., Kumar, A., Rhinehart, N., Levine, S., Shkurti, F., and Garg, A. (2020). Conservative safety critics for exploration. In *International Conference on Learning Representations*.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). Openai gym.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Chang, Y.-C., Roohi, N., and Gao, S. (2019). Neural lyapunov control. *Advances in neural information processing systems*, 32.

Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. (2021). Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097.

Chow, Y., Nachum, O., Duenez-Guzman, E., and Ghavamzadeh, M. (2018). A lyapunov-based approach to safe reinforcement learning. *Advances in neural information processing systems*, 31.

Eysenbach, B., Khazatsky, A., Levine, S., and Salakhutdinov, R. R. (2022). Mismatched no more: Joint model-policy optimization for model-based rl. *Advances in Neural Information Processing Systems*, 35:23230–23243.

Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. (2021a). D4{rl}: Datasets for deep data-driven reinforcement learning.

Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. (2021b). D4rl: Datasets for deep data-driven reinforcement learning.

Ganai, M., Gong, Z., Yu, C., Herbert, S. L., and Gao, S. (2023). Iterative reachability estimation for safe reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Ghosh, D., Ajay, A., Agrawal, P., and Levine, S. (2022a). Offline RL policies should be trained to be adaptive. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S., editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 7513–7530. PMLR.

Ghosh, D., Ajay, A., Agrawal, P., and Levine, S. (2022b). Offline rl policies should be trained to be adaptive.

Ghosh, D., Rahme, J., Kumar, A., Zhang, A., Adams, R. P., and Levine, S. (2021). Why generalization in RL is difficult: Epistemic POMDPs and implicit partial observability. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W., editors, *Advances in Neural Information Processing Systems*.

Gronauer, S. (2022). Bullet-safety-gym: A framework for constrained reinforcement learning. Technical report, mediaTUM.

Gulcehre, C., Wang, Z., Novikov, A., Paine, T., Gómez, S., Zolna, K., Agarwal, R., Merel, J. S., Mankowitz, D. J., Paduraru, C., et al. (2020). Rl unplugged: A suite of benchmarks for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:7248–7259.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR.

Janner, M., Fu, J., Zhang, M., and Levine, S. (2019). When to trust your model: Model-based policy optimization. *Advances in neural information processing systems*, 32.

Janner, M., Li, Q., and Levine, S. (2021). Offline reinforcement learning as one big sequence modeling problem. In *Advances in Neural Information Processing Systems*.

Ji, J., Zhang, B., Zhou, J., Pan, X., Huang, W., Sun, R., Geng, Y., Zhong, Y., Dai, J., and Yang, Y. (2023). Safety-gymnasium: A unified safe reinforcement learning benchmark.

Jiang, Y., Gupta, A., Zhang, Z., Wang, G., Dou, Y., Chen, Y., Fei-Fei, L., Anandkumar, A., Zhu, Y., and Fan, L. (2023). VIMA: Robot manipulation with multimodal prompts. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J., editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 14975–15022. PMLR.

Kang, K., Gradu, P., Choi, J. J., Janner, M., Tomlin, C., and Levine, S. (2022). Lyapunov density models: Constraining distribution shift in learning-based control. In *International Conference on Machine Learning*, pages 10708–10733. PMLR.

Kim, D., Lee, K., and Oh, S. (2023). Trust region-based safe distributional reinforcement learning for multiple constraints. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., and Iwasawa, Y. (2022). Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. (2020). Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33:1179–1191.

Lee, D., Han, S., Cho, T., and Lee, J. (2023). SPQR: Controlling q-ensemble independence with spiked random model for reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Lee, J., Paduraru, C., Mankowitz, D. J., Heess, N., Precup, D., Kim, K.-E., and Guez, A. (2022). Coptidice: Offline constrained reinforcement learning via stationary distribution correction estimation.

Levine, S. (2018). Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv preprint arXiv:1805.00909*.

Liu, Z., Guo, Z., Lin, H., Yao, Y., Zhu, J., Cen, Z., Hu, H., Yu, W., Zhang, T., Tan, J., and Zhao, D. (2023a). Datasets and benchmarks for offline safe reinforcement learning.

Liu, Z., Guo, Z., Yao, Y., Cen, Z., Yu, W., Zhang, T., and Zhao, D. (2023b). Constrained decision transformer for offline safe reinforcement learning.

Micheli, V., Alonso, E., and Fleuret, F. (2023). Transformers are sample-efficient world models. In *The Eleventh International Conference on Learning Representations*.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. (2022). Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Qin, Z., Chen, Y., and Fan, C. (2021). Density constrained reinforcement learning. In *International Conference on Machine Learning*, pages 8682–8692. PMLR.

Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. (2018). Improving language understanding by generative pre-training.

Rashidinejad, P., Zhu, B., Ma, C., Jiao, J., and Russell, S. (2021). Bridging offline reinforcement learning and imitation learning: A tale of pessimism. *Advances in Neural Information Processing Systems*, 34:11702–11716.

Robine, J., Höftmann, M., Uelwer, T., and Harmeling, S. (2023). Transformer-based world models are happy with 100k interactions. In *The Eleventh International Conference on Learning Representations*.

Sun, Z., Shen, Y., Zhou, Q., Zhang, H., Chen, Z., Cox, D. D., Yang, Y., and Gan, C. (2023). Principle-driven self-alignment of language models from scratch with minimal human supervision. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Tedrake, R. (2009). Underactuated robotics: Learning, planning, and control for efficient and agile machines course notes for mit 6.832. *Working draft edition*, 3(4):2.

Wang, Y., Kordi, Y., Mishra, S., Liu, A., Smith, N. A., Khashabi, D., and Hajishirzi, H. (2023). Self-instruct: Aligning language models with self-generated instructions. In Rogers, A., Boyd-Graber, J. L., and Okazaki, N., editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 13484–13508. Association for Computational Linguistics.

Xie, S. M., Raghunathan, A., Liang, P., and Ma, T. (2021). An explanation of in-context learning as implicit bayesian inference. In *International Conference on Learning Representations*.

Xu, H., Zhan, X., and Zhu, X. (2022a). Constraints penalized q-learning for safe offline reinforcement learning.

Xu, M., Shen, Y., Zhang, S., Lu, Y., Zhao, D., Tenenbaum, J., and Gan, C. (2022b). Prompting decision transformer for few-shot policy generalization. In *international conference on machine learning*, pages 24631–24645. PMLR.

Zheng, Y., Li, J., Yu, D., Yang, Y., Li, S. E., Zhan, X., and Liu, J. (2024). Safe offline reinforcement learning with feasibility-guided diffusion model. In *The Twelfth International Conference on Learning Representations*.

Checklist

1. For all models and algorithms presented, check if you include:

(a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes] Justification: Section 2 (Background) and Section 3 (Control Invariant Set, Lyapunov Model) clearly define the MDP setting, assumptions, and the proposed SAS algorithm.

(b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [No] Justification: The paper does not provide explicit computational complexity or sample complexity analysis, although iterative algorithm properties are discussed in Section 4 and Appendix F.

(c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Yes] Justification: We provide anonymized source code with dependency specifications in the supplementary material.

2. For any theoretical claim, check if you include:

(a) Statements of the full set of assumptions of all theoretical results. [Yes] Justification: Assumptions are explicitly stated in the main text (Assumption 1 in Section 4.2) and in Appendix E.

(b) Complete proofs of all theoretical results. [Yes] Justification: Full proofs are provided in Appendix F (e.g., Proof of Theorem 2, Theorem 3, and Eq. (8)).

(c) Clear explanations of any assumptions. [Yes] Justification: Explanations of assumptions (e.g., bounded Lyapunov difference) are given inline in Section 4.2 and detailed in Appendix E.

3. For all figures and tables that present empirical results, check if you include:

(a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes] Justification: We provide anonymized source code and reproduction instructions in the supplementary material; datasets (D4RL, DSRL, Safety Gymnasium) are publicly available and cited.

(b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Yes] Justification: Section C (Appendix) provides experiment setting, dataset details, normalization, and hyperparameters (Table 5).

(c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Yes] Justification: Tables 1–3 report averages over 20 evaluation episodes and 3 random seeds; normalized metrics are defined in Appendix C.2.

(d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [No] Justification: The paper does not specify the hardware or compute infrastructure.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:

(a) Citations of the creator If your work uses existing assets. [Yes] Justification: The paper cites D4RL, RL-Unplugged, DSRL, and Safety Gymnasium.

(b) The license information of the assets, if applicable. [No] Justification: Licenses for datasets are not specified.

(c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable] Justification: No new dataset or model asset is released.

(d) Information about consent from data providers/curators. [Not Applicable] Justification: Only publicly available benchmark datasets are used.

(e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable] Justification: The datasets contain only robotics and control trajectories, no sensitive content.

5. If you used crowdsourcing or conducted research with human subjects, check if you include:

(a) The full text of instructions given to participants and screenshots. [Not Applicable]

(b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]

(c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable] Justification: No human subjects or crowdsourcing are involved in this work.

Supplementary Materials: Self-alignment for Offline Safe Reinforcement Learning

A RELATED WORK

Transformer-based RL. Transformer-based RL (Janner et al., 2021; Chen et al., 2021) emerged by bridging the autoregressive pretraining paradigm of GPT (Radford et al., 2018) with offline RL on prior data. More recently, transformer-based world models such as TWM (Robine et al., 2023) and IRIS (Micheli et al., 2023) have achieved sample-efficient online RL by leveraging autoregressive imagination. Prompting in transformer-based RL has also been explored for task specification with multi-modal prompts (Jiang et al., 2023) and test-time adaptation via learned prompts (Xu et al., 2022b). CDT (Liu et al., 2023b) is closely related to our work, as it extends the Decision Transformer architecture to offline safe RL by conditioning on cost returns. Unlike these prompting and cost-conditioning approaches, SAS aligns a transformer-based world model via in-context learning by providing self-generated, Lyapunov-guided instructions, without any fine-tuning or cost annotations.

Safe RL and Lyapunov condition. Lyapunov conditions have been widely applied to safe control (Chang et al., 2019) and safe RL (Chow et al., 2018). LDM (Kang et al., 2022) integrates the Lyapunov condition with offline RL to mitigate distribution shift for safety. While safe RL is often formulated as a constrained MDP using control-theoretic surrogates—reachability (Bansal and Tomlin, 2021), iterative constraint enforcement (Ganai et al., 2023), or trust-region constraints (Kim et al., 2023)—to prevent entering unsafe regions, we instead leverage the Lyapunov condition to avoid unsafe regions caused by distribution shift (Tedrake, 2009; Bharadhwaj et al., 2020). DCRL (Qin et al., 2021) is related to our approach in that it constrains state density levels to keep the agent in high-probability states. FISOR (Zheng et al., 2024) also shares a high-level motivation with our work: both leverage a scalar stability surrogate—Hamilton–Jacobi-based feasible value in FISOR, Lyapunov-based density in SAS—to mitigate safety failures from distribution shift. However, FISOR is a *training-time* offline safe RL algorithm that learns a new diffusion-based policy from scratch, whereas SAS is a *test-time* alignment method that adapts an already-trained agent via in-context self-generated instructions without retraining. The two approaches are thus complementary: FISOR produces a safe policy during training, while SAS aligns a pretrained model at test time.

Large model alignment. Alignment techniques for LLMs aim to make pretrained models safer and more helpful, e.g., steering a general language assistant toward *helpful, honest, harmless* (HHH) behavior (Askell et al., 2021). These methods broadly fall into RLHF (Ouyang et al., 2022) and instruction-based in-context learning (Sun et al., 2023; Wang et al., 2023), the latter engineering prompts (Brown et al., 2020) or leveraging zero-shot reasoning (Kojima et al., 2022) to elicit desired outputs. In RL, aligning pretrained models for human preferences is relatively well-studied, but adapting them to unseen task specifications via demonstration (Jiang et al., 2023) or augmented prompts (Xu et al., 2022b) remains limited—even though alignment for safety is essential in real-world deployment. SAS fills this gap by adapting an offline-pretrained agent at test time through Lyapunov-guided self-generated prompts, requiring no human supervision or parameter updates.

B MODEL ARCHITECTURE

C EXPERIMENT SETTING AND HYPERPARAMETERS

C.1 Experiment setting

We conduct Hopper, Walker2d, and Humanoid in OpenAI Gym, where the agent fails and terminates when the sum of unhealthy rewards get larger. For Safety Gymnasium, we use two different robots (`Point, Car`) in 3 tasks (`Goal, Push, Button`) with two difficulties (`1,2`) respectively. In `Goal` and `Button` tasks, an agent navigate

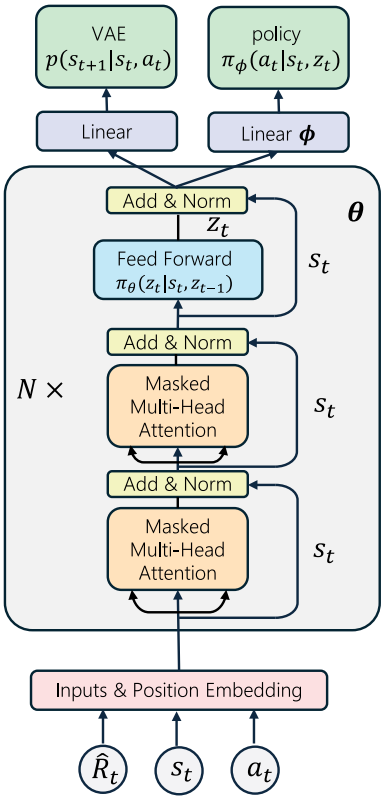

Figure 5: The architecture of decision transformer with VAE for model-based RL. The only difference with decision transformer is the additional linear layer and VAE decoder to predict the next state. We consider the output of feed-forward layer as the predictor of z_t with the parameter θ which corresponds to the high-level policy, and the combined values of s_t, z_t by the attention and residual connection are fed into the low-level policy with the linear layer ϕ.

to the goal while avoiding touching hazards, and an agent push a box to the goal in `Push` task. We denote normalized reward and cost returns as reward and cost for simplicity, and use failure in Tables for failure rate. If an agent experiences any cost due to encountering a hazard within an episode or exceeding unhealthy cost for mujoco (terminated), we considered that episode as a failure episode. The baselines we used are CDT (Liu et al., 2023b), Imitation Learning (BC-Safe, BC-All(Liu et al., 2023b; Xu et al., 2022a)), Distribution Correction Estimation (COptiDICE(Lee et al., 2022)), and Q-learning (CPQ, BCQ-Lag, BEAR-Lag(Xu et al., 2022a)).

C.1.1 Baseline Details

For completeness, we provide the full set of baseline methods used in our experiments. These methods span standard offline RL, imitation learning, distributional correction, and Q-learning approaches:

- **Constrained Decision Transformer (CDT)** (Liu et al., 2023b): A transformer-based offline RL algorithm that incorporates safety constraints into trajectory modeling.
- **Imitation Learning baselines**:
  - **BC-Safe** (Liu et al., 2023b): Behavioral cloning restricted to safe demonstrations.
  - **BC-All** (Liu et al., 2023b; Xu et al., 2022a): Behavioral cloning trained on the entire dataset, including unsafe trajectories.
- **Distribution Correction Estimation**:
  - **COptiDICE** (Lee et al., 2022): A distribution correction approach that optimizes cost-sensitive objectives for safe offline RL.

- **Q-learning baselines**:
  - **CPQ** (Xu et al., 2022a): Constrained Policy Q-learning for safe offline RL.
  - **BCQ-Lag** (Xu et al., 2022a): Offline BCQ augmented with a Lagrangian penalty for safety.
  - **BEAR-Lag** (Xu et al., 2022a): Offline BEAR similarly extended with Lagrangian safety regularization.

We adopt standard hyperparameters for these baselines from their respective papers. All implementations are based on publicly available code where possible.

C.2 Hyperparameters for the Experiments

During the training of Decision Transformer, we applied warmup for the first 10000 steps, and we used the ReLU activation function. Further details about the hyperparameters can be found in Table 3.

Table 3: Hyperparameters for the experiments

Common Parameters	Safety-Gymnasium
Action hidden size	[256, 256] for all methods except CDT, DT
VAE hidden size	[400, 400] BCQ-Lag, BEAR-Lag, CPQ
Cost thresholds	[20, 40, 80]
Gradient steps	100000
$[K_{\mathcal{P}}, K_{\mathcal{I}}, K_{\mathcal{D}}]$	[0.1, 0.003, 0.001] BCQ-Lag, BEAR-Lag
Batch size	512
Actor learning rate	0.0001
Critic learning rate	0.001

Parameters	CDT	DT
Number of layers	3	3
Number of attention heads	8	1
Embedding dimension	128	128
Batch size	2048	64
Context length K	300	20
Learning rate	0.0001	0.0001
Dropout	0.1	0.1
Adam betas	(0.9, 0.999)	(0.9, 0.999)

C.3 Normalized Score

We applied normalization to both reward return and cost return to make it easier to compare for all environments. Let $r_{max}(\mathcal{M})$ and $r_{min}(\mathcal{M})$ denote the maximum reward return and minimum reward return in the dataset $\mathcal{T}$, respectively. Then, the normalized reward return is computed as:

$$R_{normalized} = \frac{R_\pi - r_{min}(\mathcal{M})}{r_{max}(\mathcal{M}) - r_{min}(\mathcal{M})}$$

where $\mathcal{R}_\pi$ denotes the evaluated reward return obtained by the agent. Normalized cost return is defined as the ratio between the cost return obtained by the agent and the target cost κ:

$$C_{normalized} = \frac{C_\pi + \epsilon}{\kappa + \epsilon}$$

where ϵ is a small positive number for numerical stability. The values are averaged across three different cost thresholds, 20 evaluation episodes, and three random seeds.

C.4 Dataset Details

We conducted experiments using the OpenAI Gym's medium and expert datasets from https://github.com/Farama-Foundation/D4RL and the Safety Gymnasium's expert dataset from https://github.com/liuzuxin/OSRL/tree/main. Detailed information about the dataset is presented in Appendix C.4. The Max Cost means the maximum cost return in dataset trajectories.

D COMPARISON WITH FISOR

As discussed in Appendix A, FISOR (Zheng et al., 2024) is a training-time offline safe RL method, while SAS is a test-time alignment approach. Although the two methods address different problem settings, we provide a direct empirical comparison for completeness. Table 5 reports normalized reward and cost on overlapping environments

Table 4: Dataset details

Benchmark	Task	Max Timestep	Action Space	State Space	Max Cost	Trajectories
Safety Gymnasium	SafetyPointGoal1-v0	1000	2	60	100	2022
	SafetyPointGoal2-v0			60	200	3442
	SafetyPointPush1-v0			76	150	2379
	SafetyPointPush2-v0			76	200	3242
	SafetyPointButton1-v0			76	200	2268
	SafetyPointButton2-v0			76	250	3288
	SafetyCarGoal1-v0			72	200	1671
	SafetyCarGoal2-v0			72	250	4105
	SafetyCarPush1-v0			88	250	2871
	SafetyCarPush2-v0			88	400	4407
	SafetyCarButton1-v0			88	250	2656
	SafetyCarButton2-v0			88	300	3755

from Safety-Gymnasium and Bullet-Safety-Gym. CDT+SAS generally achieves higher reward but incurs higher cost, reflecting its performance-oriented design. FISOR consistently yields lower cost at the expense of some reward, reflecting its safety-oriented training objective. The two approaches are complementary: FISOR can be used to obtain a safe policy during training, while SAS can further align a pretrained agent at deployment.

Table 5: Comparison of CDT+SAS and FISOR on overlapping environments. Reward (↑) and Cost (↓) are normalized. Values below 1.0 for cost indicate safe behavior.

	FISOR		CDT+SAS	
Task	Reward	Cost	Reward	Cost
Safety-Gymnasium				
PointGoal1	0.30	**0.35**	**0.65**	1.27
PointGoal2	0.30	**0.35**	**0.52**	0.94
PointPush1	**0.49**	0.83	0.26	**0.54**
PointPush2	**0.44**	**0.11**	0.20	0.53
PointButton1	0.01	**0.58**	**0.51**	1.27
PointButton2	0.01	**0.58**	**0.41**	0.98
Bullet-Safety-Gym				
BallRun	**0.18**	**0.00**	0.04	0.29
CarRun	**0.73**	**0.14**	0.72	0.39
DroneRun	0.30	**0.55**	**0.38**	0.78
AntRun	**0.45**	**0.03**	0.29	0.45
BallCircle	**0.34**	**0.00**	0.31	0.47
CarCircle	**0.40**	**0.11**	0.22	0.75
DroneCircle	0.48	**0.00**	**0.54**	0.63
AntCircle	0.20	**0.00**	**0.26**	0.34

E ABLATION STUDIES

E.1 Number of trajectories sampled for imagination

We employ Decision Transformer to imagine multiple trajectories under both condition $\mathcal{U}_t$ and condition $\mathcal{V}_t$. In our case, we sampled 5 trajectories for each condition $\mathcal{U}_t$ and $\mathcal{V}_t$. As part of an ablation study, we compared the results of sampling 100 trajectories in experiment with our experimental results. The experiment revealed that there was not a significant difference in the model's performance due to the difference in the number of sampled trajectories. In PointGoal1 environment, an increase in cost was observed when the number of sampled trajectories was 100.

E.2 Time step length to calculate E

We calculated and approximated E from trajectories imagined by Decision Transformer under both conditions U and V. We conducted experiments with a default time step length of 3 for computing E. As part of an ablation study, we also experimented with time step lengths of 1 and 10, comparing the results with our findings. In the results for the CarGoal1 environment, the cost is lowest when the time step length is 10, while in the PointGoal1 environment, it is actually highest. This indicates that increasing the time step length for calculating E does not noticeably improve the model's performance.

E.3 Time step length of trajectory in prompt

We extracted the trajectory from a specific time t to 5 time steps before that from trajectories generated through Condition $\mathcal{U}_t$ and $\mathcal{V}_t$. We then fed this truncated trajectory into the prompt of Decision Transformer at the test time. As part of an ablation study, we experimented with the time step length of Decision Transformer's prompt, setting it to 3 and 10. For each time step length of the prompt (3, 5, 10), there are instances where the cost in the experimental results is the highest, as well as instances where it is the lowest. Hence, it can be concluded that the time step length of the prompt does not significantly impact the model's performance.

E.4 Number of prompts

We proceeded by using a single trajectory fragment generated by our algorithm as the prompt for Decision Transformer. As part of an ablation study, we compared the performance of our approach with the method of concatenating three or five trajectory fragments obtained by running our algorithm three or five times, respectively, and using them as a prompt. The experimental results show that the method of using five trajectory fragments as a prompt resulted in higher costs. While there is some difference in the PointPush2 environment when the number of fragments is 1 or 3, overall, the performance fluctuates without a clear trend.

E.5 Model size of Decision Transformer

We conducted experiments to observe how the effectiveness of SAS varies with the model size of the Decision Transformer. Starting from the smallest size, the default Decision Transformer, we experimented with sizes ranging from GPT-mini to larger sizes like GPT2. When examining the PointGoal1 environment, it seems that as the model size increases, the cost also tends to increase. However, looking at the PointPush2 environment, the opposite trend is observed, where the model with the smallest size has the highest cost, suggesting that there may not be a significant correlation. However, concerning failures, except for the gpt-mini size in PointGoal1, it can be observed that as the model size increases, failures generally decrease.

F COMPLETE EXPERIMENT RESULTS

F.1 Results for all the datasets

We present the results for a total of 16 datasets in Table 7. These results include an additional experiment on four Circle tasks (PointCircle1, PointCircle2, CarCircle1, CarCircle2) and eight tasks in bullet-safety-gym environment(BallRun, CarRun, DroneRun, AntRun, BallCircle, CarCircle, DroneCircle, AntCircle). In PC2, CC1,

Table 6: Ablation study in the Safety Gymnasium. DT+rand involves inserting a random trajectory into the prompt, and DT+maxmax includes the trajectory with the argmax of the maximum value of E as the prompt. **Bold**: the smallest cost among the four models. **Blue**: DT+SAS has a lower failure rate than DT. **Red**: DT+SAS has a higher cost than DT but the reward is also higher.

Environment		PG1	PG2	PP1	PP2	PB1	PB2	CG1	CG2	CP1	CP2	CB1	CB2
DT	reward	0.660	0.377	0.218	0.202	0.379	0.495	0.638	0.513	0.35	0.204	0.237	0.212
	cost	1.319	2.625	0.927	0.782	**1.188**	1.309	0.976	1.466	0.678	1.174	1.419	1.045
	failure	0.883	1.000	0.667	0.875	0.950	0.983	0.917	0.925	0.667	0.950	0.950	0.950
DT+SAS(ours)	reward	0.655	0.650	0.283	0.242	**0.485**	0.508	0.666	0.483	0.307	0.218	0.174	0.138
	cost	1.185	**1.783**	**0.622**	**0.639**	1.375	1.205	**0.846**	**1.148**	**0.513**	**1.158**	**1.083**	**0.836**
	failure	**0.867**	**0.983**	0.767	**0.850**	0.950	**0.967**	**0.867**	**0.850**	**0.483**	**0.900**	0.975	1.000
DT+rand	reward	0.665	0.587	0.303	0.240	0.445	0.462	0.672	0.507	0.311	0.230	0.175	0.111
	cost	1.258	1.811	0.678	0.758	1.485	**0.960**	1.002	1.438	0.549	1.341	1.259	0.963
	failure	0.900	1.000	0.767	0.875	1.000	0.950	0.867	0.975	0.617	0.950	0.975	0.925
DT+maxmax	reward	0.644	0.521	0.271	0.200	0.486	0.441	0.636	0.512	0.321	0.206	0.131	0.126
	cost	**0.990**	2.152	0.640	0.730	1.808	1.273	1.034	1.497	0.574	1.271	1.103	0.911
	failure	0.775	1.000	0.767	0.783	1.000	0.950	0.817	0.933	0.625	0.975	0.967	0.975

and CC2 environments, CDT+SAS exhibited the highest reward among safe agents. CDT+SAS demonstrates lower costs than CDT in all four environments.

Table 7: Complete evaluation results of the baselines and the Decision Transformer with our method (DT+SAS) and Constrained Decision Transformer with our method (CDT+SAS) in the Safety Gymnasium environment. The values are averaged across three different cost thresholds, 20 evaluation episodes, and three random seeds. Gray: Unsafe agents. **Bold**: Safe agents whose normalized cost is less than 1. **Blue**: Agents which has highest reward among safe agents

Task	DT + SAS		CDT + SAS		CDT		BC-All		BC-Safe		BCQ-Lag		BEAR-Lag		CPQ		COptiDICE	
	reward	cost	reward	cost	reward	cost	reward	cost	reward	cost	reward	cost	reward	cost	reward	cost	reward	cost
PointGoal1	0.66	1.19	0.65	1.27	0.69	1.12	**0.65**	**0.95**	**0.43**	**0.54**	**0.71**	**0.98**	0.74	1.18	**0.57**	**0.35**	0.49	1.66
PointGoal2	0.65	1.78	**0.52**	**0.94**	0.59	1.34	0.54	1.97	**0.29**	**0.78**	0.67	3.18	0.67	3.11	0.4	1.31	0.38	1.92
PointPush1	**0.28**	**0.62**	**0.26**	**0.54**	**0.24**	**0.48**	**0.19**	**0.61**	**0.13**	**0.43**	**0.33**	**0.86**	**0.22**	**0.79**	**0.2**	**0.83**	**0.13**	**0.83**
PointPush2	**0.24**	**0.64**	**0.20**	**0.53**	**0.21**	**0.65**	**0.18**	**0.91**	**0.11**	**0.8**	**0.23**	**0.99**	0.16	0.89	0.11	1.04	0.02	1.18
PointButton1	0.49	1.38	0.51	1.27	0.5	1.68	0.1	10.5	**0.06**	**0.52**	0.24	1.73	0.2	1.6	0.69	3.2	0.13	1.4
PointButton2	0.51	1.14	**0.41**	**0.98**	0.46	1.57	0.27	2.02	0.16	1.1	0.4	2.66	0.43	2.47	0.58	4.3	0.15	1.51
PointCircle1	0.69	1.81	**0.54**	**0.21**	**0.59**	**0.69**	0.79	3.98	**0.41**	**0.16**	0.54	2.38	0.73	3.28	**0.43**	**0.75**	0.86	5.51
PointCircle2	0.42	1.69	**0.63**	**0.47**	0.64	1.05	0.66	4.17	**0.48**	**0.99**	0.66	2.6	0.63	4.27	0.24	3.58	0.85	8.61
CarGoal1	**0.67**	**0.85**	**0.65**	**0.90**	0.66	1.21	**0.39**	**0.33**	**0.24**	**0.28**	**0.47**	**0.78**	0.61	1.13	0.79	1.42	**0.35**	**0.54**
CarGoal2	0.48	1.15	**0.42**	**0.98**	0.48	1.25	0.23	1.05	**0.14**	**0.51**	0.3	1.44	0.28	1.01	0.65	3.75	**0.25**	**0.91**
CarPush1	**0.31**	**0.51**	**0.31**	**0.49**	**0.31**	**0.4**	**0.22**	**0.36**	**0.14**	**0.33**	**0.23**	**0.43**	**0.21**	**0.54**	**-0.03**	**0.95**	**0.23**	**0.5**
CarPush2	0.22	1.16	**0.21**	**0.75**	0.19	1.3	**0.14**	**0.9**	**0.05**	**0.45**	0.15	1.38	0.1	1.2	0.24	4.25	0.09	1.07
CarButton1	0.17	1.08	**0.27**	**0.98**	0.21	1.6	0.03	1.38	**0.07**	**0.85**	0.04	1.63	0.18	2.72	0.42	9.66	-0.08	1.68
CarButton2	**0.14**	**0.84**	0.30	1.11	0.13	1.58	-0.13	1.24	**-0.01**	**0.63**	0.06	2.13	-0.01	2.29	0.37	12.51	-0.07	1.59
CarCircle1	0.41	1.84	**0.47**	**0.52**	0.6	1.73	0.72	4.39	0.37	1.38	0.73	5.25	0.76	5.46	0.02	2.29	0.7	5.72
CarCircle2	0.63	1.69	**0.56**	**0.62**	0.66	2.53	0.76	6.44	0.54	3.38	0.72	6.58	0.74	6.82	0.44	2.69	0.77	7.99
BallRun	0.99	1.6	**0.04**	**0.29**	0.39	1.16	0.6	5.08	0.27	1.46	0.76	3.91	-0.47	5.03	0.22	1.27	0.59	3.52
CarRun	8.12	1.06	**0.72**	**0.39**	**0.99**	**0.65**	**0.97**	**0.33**	**0.94**	**0.22**	**0.94**	**0.15**	0.68	7.78	0.95	1.79	**0.87**	**0**
DroneRun	0.76	1.58	**0.33**	**0.78**	**0.63**	**0.79**	0.24	2.13	**0.28**	**0.74**	0.72	5.54	0.42	2.47	0.33	3.52	0.67	4.15
AntRun	1.08	2.43	**0.32**	**0.14**	**0.72**	**0.91**	0.72	2.93	0.65	1.09	0.76	5.11	**0.15**	**0.73**	**0.03**	**0.02**	**0.61**	**0.94**
BallCircle	0.81	1.41	**0.32**	**0.38**	0.77	1.07	0.74	4.71	**0.52**	**0.65**	0.69	2.36	0.86	3.09	**0.64**	**0.76**	0.7	2.61
CarCircle	0.85	1.76	**0.19**	**0.22**	**0.75**	**0.95**	0.58	3.74	**0.5**	**0.84**	0.63	1.89	0.74	2.18	**0.71**	**0.33**	0.49	3.14
DroneCircle	0.82	1.55	**0.51**	**0.42**	**0.63**	**0.98**	0.72	3.03	**0.56**	**0.57**	0.8	3.07	0.78	3.68	-0.22	1.28	0.26	1.02
AntCircle	0.59	1.18	**0.26**	**0.34**	0.54	1.78	0.58	4.9	**0.4**	**0.96**	0.58	2.87	0.65	5.48	**0**	**0**	0.17	5.04

F.2 Additional comparison with SOTA offline RL methods and offline meta-RL

We note that our DT+SAS which uses the pretrained DT without cost training data outperforms the above SOTA offline safe RL methods. Moreover, we provide the comparison with CQL, SAC-n, and APE-V which is the online (few-shot) adaptation method for the offline RL algorithm in the table below. We note that our method shows the better improvement compared to the reported value of SAC-n → APE-V in APE-V paper. However, the target task of offline meta-RL focuses on the adaptation performance when the goal of the target

Table 8: The modified version of Table 6 with standard deviation across 3 cost thresholds, 20 evaluation episodes, and 3 random seeds.

Task	CDT				CDT+SAS				DT				DT+SAS			
	reward		cost		reward		cost		reward		cost		reward		cost	
	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std	mean	std
PointGoal1	0.69	0.007	1.12	0.037	0.65	0.007	1.27	0.062	0.66	0.02	1.32	0.31	0.66	0.03	**1.19**	0.15
PointGoal2	0.59	0.017	1.34	0.054	0.52	0.036	**0.94**	0.158	0.38	0.02	2.63	0.05	0.65	0.09	1.78	0.17
PointPush1	0.24	0.012	0.48	0.023	0.26	0.027	0.54	0.019	0.22	0.06	0.93	0.21	0.28	0.01	**0.62**	0.10
PointPush2	0.21	1.363	0.65	31.063	0.20	0.038	**0.53**	0.089	0.20	0.08	0.78	0.45	0.24	0.06	**0.64**	0.09
PointButton1	0.5	0.006	1.68	0.049	0.51	0.026	**1.27**	0.044	0.38	0.04	1.19	0.18	0.49	0.05	1.38	0.21
PointButton2	0.46	0.019	1.57	0.047	0.41	0.019	**0.98**	0.026	0.50	0.06	1.31	0.14	0.51	0.00	**1.14**	0.13
CarGoal1	0.66	0.008	1.21	0.057	0.65	0.008	**0.90**	0.035	0.64	0.02	0.98	0.12	0.67	0.03	**0.85**	0.16
CarGoal2	0.48	0.032	1.25	0.095	0.42	0.032	**0.98**	0.047	0.51	0.04	1.47	0.32	0.48	0.03	**1.15**	0.20
CarPush1	0.31	0.018	0.40	0.068	0.31	0.018	**0.49**	0.097	0.35	0.07	0.68	0.22	0.31	0.01	**0.51**	0.15
CarPush2	0.19	0.022	1.30	0.081	0.21	0.023	**0.75**	0.120	0.20	0.03	1.17	0.26	0.22	0.01	**1.16**	0.26
CarButton1	0.21	0.014	1.60	0.106	0.27	0.081	**0.98**	0.006	0.24	0.04	1.42	0.04	0.17	0.03	**1.08**	0.17
CarButton2	0.13	0.031	1.58	0.034	0.30	0.009	**1.11**	0.025	0.21	0.04	1.05	0.21	0.14	0.03	**0.84**	0.08

task changes significantly, which differs critically from measuring the generalization performance that is the aim of our paper, making it challenging to conduct additional experiments.

Table 9: Experiment results with CQL algorithm (Kumar et al., 2020) and APE-V algorithm (Ghosh et al., 2022b) in D4RL (Fu et al., 2021a) datasets.

Task Name	CQL	DT		DT+SAS			SAC-N	APE-V	
	reward	reward	failure	reward	failure	improve(%)	reward	reward	improve(%)
hopper-medium-expert	96.9	111.8	0.1	110.4	0.05	-1.25	110	105.7	-3.91
hopper-medium-replay	86.3	94.3	0	97.3	0	3.18	101.8	98.5	-3.24
walker2d-medium-expert	109.1	108.3	0	107.5	0	-0.74	116	110	-5.17
walker2d-medium-replay	76.8	43.9	1	69.1	0.6	57.4	78.7	82.9	5.34

G PROOF

We first provide technical results in the main paper. By defining the condition probability of prompt $p_{\mathbf{1:K}}$ given high-level policy π_θ^{high}, we leverage $r_K(\theta)$ to make sure that the well-designed prompt is selected when it is from underlying the safe high-level policy $\pi_{\theta^*}^{\text{high}}$. In details, the length variable K can be composed of two conditions, the length of prompt and the number of prompt. We note that we can have high probability of $p(\mathcal{O}_t = 1|\mathbf{z}_t) = \exp(r(\pi_\theta^{\text{high}}))$ when we provide the most matching prompt $\mathbf{p}^*$ with the underlying $\pi_{\theta^*}^{\text{high}}$.

G.1 Connection of density and safe RL

Let U be the universal set of state-action space and B be the set $\{(s,a)|(s,a), C(s,a) < C_{th}\}$ where C_{th} is the some threshold of cost which satisfies $C_{th} \leq d(1-\gamma)$. Assume that the cost value is bounded as $0 \leq C(s,a) \leq C_{max}$. We define a volume constant $\alpha = \mathbb{E}_{(s,a)\sim U}[\mathbf{1}((s,a) \in B)]$, which is $0 < \alpha < 1$. Then, we can write the above inequality as

$$\begin{aligned} J_C(\pi) - d &= \mathbb{E}_{(s,a)\sim B}\left[\hat{\rho}^\pi(s,a)C(s,a)\right] - \alpha d \\ &\quad + \mathbb{E}_{(s,a)\sim B^C}\left[\hat{\rho}^\pi(s,a)C(s,a)\right] - (1-\alpha)d \\ &\leq \mathbb{E}_{(s,a)\sim B}\left[\hat{\rho}^\pi(s,a)C_{th} - d\right] \\ &\quad + \mathbb{E}_{(s,a)\sim B^C}\left[\hat{\rho}^\pi(s,a)C_{max} - d\right]. \end{aligned} \tag{11}$$

We can observe that expert policies in the offline RL dataset $\mathcal{D}$ should have low values in B^C to satisfy the constraint inequality of Equation (11) less than zero. Now, we know that reducing the marginal value of occupancy

measure over B^C leads to the lower bound of the cost function of π. Suppose that $\hat{\rho}^\pi(s,a) \leq \frac{d}{C_{max}}$ for $(s,a) \in B^C$. We consider the definition of occupancy measure and assume that occupancy measure is a continuous function. We have $\frac{d}{C_{max}} \leq \hat{\rho}^\pi(s,a) \leq \frac{d}{C_{th}} \leq \frac{1}{1-\gamma}$ for $(s,a) \in B$, and then get $J_C(\pi) - d \leq 0$. This is an intuitive condition to be an expert policy trained by the constrained RL in Equation (3).

G.2 Proof of Equation (8)

To show the derivation, we start from

$$p(\tau \mid p_{1:K}, s_1^{\text{test}}) = \int_\theta p(\tau \mid p_{1:K}, s_1^{\text{test}}, \theta)\, p(\theta)\, d\theta. \tag{12}$$

To check the optimality between the generated trajectory and the prompt, we prove

$$p(\mathcal{O}_{\text{traj}} \mid p_{1:K}, s_1^{\text{test}}) = \int_\theta \sum_{z_1^{\text{test}} \in \mathcal{Z}} \left(g_{\pi_\theta}(\tau, z_1^{\text{test}}) \prod_{t=1} p(\mathcal{O}_t \mid s_t^{\text{test}}, a_t^{\text{test}}) \right) p(z_1^{\text{test}} | \mathbf{p}_{1:K}, s_1^{\text{test}}, \theta) e^{K \cdot r_K(\theta)}\, p(\theta)\, d\theta, \tag{13}$$

where

$$\sum_{z_1^{\text{test}} \in \mathcal{Z}} \prod_{t=1} p(s_{t+1}^{\text{test}} \mid s_t^{\text{test}}, a_t^{\text{test}}) \underbrace{p(a_t^{\text{test}} \mid s_t^{\text{test}}, z_t^{\text{test}})}_{\pi_\phi^{\text{low}}} \underbrace{p_\theta(z_t^{\text{test}} \mid s_t^{\text{test}}, z_{t-1}^{\text{test}})}_{\pi_\theta^{\text{high}}} =: \sum_{z_1^{\text{test}} \in \mathcal{Z}} g_{\pi_\theta}(\tau, z_1^{\text{test}}). \tag{14}$$

By Bayes' rule and the law of total probability, we have

$$\begin{aligned}
p(\mathcal{O}_{\text{traj}} \mid p_{1:K}, s_1^{\text{test}}) &= \int_\theta p(\tau \mid p_{1:K}, s_1^{\text{test}}, \theta)\; p(\theta \mid p_{1:K}, s_t^{\text{test}})\, d\theta \\
&\propto \int_\theta p(\tau \mid p_{1:K}, s_1^{\text{test}}, \theta)\; p(p_{1:K}, s_t^{\text{test}} \mid \theta)\, p(\theta)\, d\theta \\
&= \int_\theta \sum_{z_1^{\text{test}} \in \mathcal{Z}} \left(g_{\pi_\theta}(\tau, z_1^{\text{test}}) \prod_{t=1} p(\mathcal{O}_t \mid s_t^{\text{test}}, a_t^{\text{test}}) \right) p(z_1^{\text{test}} | \mathbf{p}_{1:K}, s_1^{\text{test}}, \theta) \frac{p(p_{1:K}, s_t^{\text{test}} \mid \theta)}{p(p_{1:K}, s_t^{\text{test}} \mid \theta^*)}\, p(\theta)\, d\theta \\
&= \int_\theta \sum_{z_1^{\text{test}} \in \mathcal{Z}} \left(g_{\pi_\theta}(\tau, z_1^{\text{test}}) \prod_{t=1} p(\mathcal{O}_t \mid s_t^{\text{test}}, a_t^{\text{test}}) \right) p(z_1^{\text{test}} | \mathbf{p}_{1:K}, s_1^{\text{test}}, \theta) \exp\big(K \cdot r_K(\theta)\big)\, p(\theta)\, d\theta.
\end{aligned} \tag{15}$$

By the definition of $r_K(\theta)$, under distinguishability for all $\pi_\theta^{\text{high}} \neq \pi_{\theta^*}^{\text{high}}$, $r_K(\theta)$ converges to a negative constant, and letting $K \to \infty$ gives $\exp(r_K(\pi_\theta^{\text{high}})) = 0$ for all $\pi_\theta^{\text{high}} \neq \pi_{\theta^*}^{\text{high}}$, and $\exp(r_K(\pi_\theta^{\text{high}})) = 1$ for $\pi_\theta^{\text{high}} = \pi_{\theta^*}^{\text{high}}$. A detailed discussion of distinguishability can be found in Xie et al. (2021).

Moreover, the graphical model contains $p(z_t \mid s_t, z_{t-1})$, which samples latent skill variables when s_t is given. Thus, the transformer with latent variables can be viewed as hierarchical RL with high-level policy $\pi_\theta^{\text{high}} = p(z_t \mid s_t, z_{t-1})$.

Hence, interpreting the transformer as implicit Bayesian inference of in-context learning (Xie et al., 2021), we obtain a safe high-level policy when the sampled trajectory instruction in Algorithm 1 satisfies Lyapunov conditions at every step. Consequently, the in-context learner can act at the test-time initial state according to a Lyapunov-stable policy.

G.3 Proof of Theorem 2

Since $\mathcal{U}_t$ and $\mathcal{V}_t$ are both optimality variables indicating the Lyapunov condition, we apply probabilistic inference for RL as

$$\begin{aligned}
\log p(\mathcal{U}_{1:T}, \mathcal{V}_{1:T} \mid \tau) &= \log \left(p(s_1) \prod_{t=1}^{T} p(\mathcal{U}_t, \mathcal{V}_t \mid s_t, a_t)\, p(s_{t+1} \mid s_t, a_t)\, p(a_t \mid s_t, z_t)\, p(z_t \mid s_t, z_{t-1}) \right) \\
&= \sum_{t=1}^{T} \log p(\mathcal{U}_t, \mathcal{V}_t \mid s_t, a_t) + C.
\end{aligned} \tag{16}$$

When all $\mathcal{U}_t = \mathcal{V}_t = 1$, the trajectory perfectly satisfies the Lyapunov condition.

Recall that a trajectory is asymptotically stable if

$$\textbf{(1)}\ G(s_e, a_e) = 0, \quad \textbf{(2)}\ G(s_t, a_t) > 0,\ \forall (s_t, a_t) \neq (s_e, a_e), \quad \textbf{(3)}\ G(s_t, a_t) \geq G(s_{t+1}, a_{t+1}). \tag{17}$$

We define our Lyapunov function as

$$G_{\text{SAS}}(s_t, a_t) = \min_{\pi} \max_{t'} E(s_{t'}, \pi(s_{t'})) - E(s_t, a_t), \tag{18}$$

so that the equilibrium point satisfies $G_{\text{SAS}}(s_e, a_e) = 0$. Thus, $\mathcal{U}_t = 1$ corresponds to condition (2), and $\mathcal{V}_t = 1$ corresponds to condition (3).

If we choose the distributions

$$p(\mathcal{U}_t = 1 \mid s_t, a_t) \propto \exp(\mathbf{1}[\, G_{\text{SAS}}(s_t, a_t) > 0\,])\,, \tag{19}$$
$$p(\mathcal{V}_t = 1 \mid s_t, a_t) \propto \exp(\mathbf{1}[\, G_{\text{SAS}}(s_t, a_t) - G_{\text{SAS}}(s_{t+1}, a_{t+1}) \geq 0\,])\,, \tag{20}$$

then

$$\begin{aligned}\sum_{t=1}^{T} \log p(\mathcal{U}_t, \mathcal{V}_t \mid s_t, a_t) &= \sum_{t=1}^{T} \log p(\mathcal{V}_t \mid s_t, a_t, \mathcal{U}_t)\, p(\mathcal{U}_t \mid s_t, a_t) \\ &\propto \sum_{t=1}^{T} \mathbf{1}[\, G_{\text{SAS}}(s_t, a_t) > 0\,] + \sum_{t=1}^{T} \mathbf{1}[\, G_{\text{SAS}}(s_t, a_t) - G_{\text{SAS}}(s_{t+1}, a_{t+1}) \geq 0\,].\end{aligned} \tag{21}$$

Maximizing the above implies that the trajectory approaches the Lyapunov condition.

G.4 Proof of Theorem 4

The goal of our method is to keep occupancy measures in the distribution of the target control-invariant set $\mathcal{R} = \{(s_t, a_t) | c_1 \leq E(s_t, a_t) \leq c_2\}$ where $E(s_t, a_t) = -\log \rho(s_t, a_t)$ for utilizing the pretrained expert distribution. As our Lyapunov function approximation is defined as

$$G(s_t, a_t) = \min_{i=1,\cdots,N} \max_{j=1,\cdots T} E(s_j, \pi_i(s_j)) - E(s_t, a_t)$$

for N sample trajectories with the episode length T in the first loop of Algorithm 1. Suppose that c_2 is some constant that is larger than $\min_{i=1,\cdots,N} \max_{j=1,\cdots T} E(s_j, \pi_i(s_j))$ for any N, T. For clarity, we assume that the state–action pairs $\{(s_t, a_t)\}_{t=1}^{T}$ are sampled i.i.d. from the dataset $\mathcal{D}$. In practice, trajectories in an MDP exhibit temporal dependence; however, the same arguments extend under standard β-mixing or ergodicity conditions, which yield similar bounds up to constant factors. The proof proceeds in two steps:

1. **Bounding high-energy events.** Using Markov's inequality, we control the probability that a sampled trajectory includes pairs with energy $E(s, a) = -\log \hat{\rho}(s, a)$ exceeding the threshold c_2. This yields the first term in equation 10.

2. **Ensuring Lyapunov monotonicity.** Applying Hoeffding's inequality to the Lyapunov descent indicators $\{\mathcal{V}_t\}$, we bound the probability that cumulative descent violations exceed $\kappa(c_2 - c_1)/L$. This leads to the second exponential term.

Combining both bounds yields the probability estimate in equation 10. We now demonstrate that Algorithm 1 reduces the probability of escaping from the control invariant set as the numbers of iterations, N and M for the first and second loops, respectively, increase.

Assumption 5. The difference $\|G(s_t, a_t) - G(s_{t+1}, a_{t+1})\|$ in Eq. (3) over the transition $\mathcal{T}$ is bounded as $\|G(s_t, a_t) - G(s_{t+1}, a_{t+1})\| \leq L$ for all t.

Proof. First note that $P[\tau \not\subset \mathcal{R}]$ is less than the sum of the probability of $E(s_t, a_t) \geq c_2$ for all data points in N trajectories and the probability that all M trials moves below $E(s_t, a_t) \leq c_1$. By using Markov's inequality for the first term of RHS and Hoeffding's inequality for the second term of RHS. Then, we have

$$\begin{aligned} P[\tau_{\text{best}} \not\subset \mathcal{R}] &\leq (P[E(s,a) \geq c_2])^{NT} + \left(P\left[\sum_{t=1}^{T} \mathbf{1}(\mathcal{V}_t \neq 1) \geq \frac{\kappa(c_2 - c_1)}{L} \right] \right)^M \\ &\leq \left[\frac{\mathbb{E}_{(s,a)\sim\mathcal{D}}[E(s,a)]}{c_2} \right]^{NT} + \exp\left(-\frac{2M\kappa^2(c_2 - c_1)^2}{TL^2} \right) \end{aligned}$$

for some constant κ to describe the average distance to escape.

We note that the probability of $E(s_t, a_t) \geq c_2$ for all data points in N trajectories implies $\mathbf{1}(\mathcal{U}_t \neq 1)$ because $G(s_t, a_t) = \min_{i=1,\cdots,N} \max_{j=1,\cdots T} E(s_j, \pi_i(s_j)) - E(s_t, a_t) < c_2 - E(s_t, a_t) < 0$.

□