Dynamic Model Predictive Shielding for Provably Safe Reinforcement Learning

Arko Banerjee
The University of Texas at Austin
arko.banerjee@utexas.edu

Kia Rahmani
The University of Texas at Austin
kia@durable.ai

Joydeep Biswas
The University of Texas at Austin
joydeepb@utexas.edu

Isil Dillig
The University of Texas at Austin
isil@utexas.edu

Abstract

Among approaches for provably safe reinforcement learning, Model Predictive Shielding (MPS) has proven effective at complex tasks in continuous, high-dimensional state spaces, by leveraging a *backup policy* to ensure safety when the learned policy attempts to take unsafe actions. However, while MPS can ensure safety both during and after training, it often hinders task progress due to the conservative and task-oblivious nature of backup policies. This paper introduces Dynamic Model Predictive Shielding (DMPS), which optimizes reinforcement learning objectives while maintaining provable safety. DMPS employs a local planner to dynamically select safe recovery actions that maximize both short-term progress as well as long-term rewards. Crucially, the planner and the neural policy play a synergistic role in DMPS. When planning recovery actions for ensuring safety, the planner utilizes the neural policy to estimate long-term rewards, allowing it to *observe* beyond its short-term planning horizon. Conversely, the neural policy under training learns from the recovery plans proposed by the planner, converging to policies that are both *high-performing* and *safe* in practice. This approach guarantees safety during and after training, with bounded recovery regret that decreases exponentially with planning horizon depth. Experimental results demonstrate that DMPS converges to policies that rarely require shield interventions after training and achieve higher rewards compared to several state-of-the-art baselines.

1 Introduction

Safe Reinforcement Learning (SRL) [1, 2] aims to learn policies that adhere to important safety requirements and is essential for applying reinforcement learning to safety-critical applications, such as autonomous driving. Some SRL approaches give *statistical* guarantees and reduce safety violations by directly incorporating safety constraints into the learning objective [3, 4, 5, 6]. In contrast, *Provably Safe* RL (PSRL) methods aim to learn policies that *never* violate safety and are essential in high-stakes domains where even a single safety violation can be catastrophic [7].

A common approach to PSRL is *shielding* [8], where actions proposed by the policy are monitored for potential safety violations. If necessary, these actions are overridden by a safe action that is guaranteed to retain the system's safety. Model Predictive Shielding (MPS) is an emerging PSRL method that has proven effective in high-dimensional, continuous state spaces with complex dynamics, surpassing previous shielding approaches in applicability and performance [9, 10]. At a high level, MPS methods leverage the concept of *recoverable states* that lead to a safe equilibrium in N time-steps by following

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

a so-called *backup policy*. MPS approaches dynamically check (via simulation) if a proposed action results in a recoverable state and follow the backup policy if it does not.

However, an important issue with existing MPS approaches is that the backup policies are not necessarily aligned with the primary task's objectives and often propose actions that, while safe, impede progress towards task completion—even when alternative safe actions are available that do not obstruct progress in the task. Intuitively, this occurs because the backup policy is designed with the sole objective of driving the agent to the closest equilibrium point rather than making task-specific progress. For instance, in an autonomous navigation task, if the trained policy suggests an action that could result in a collision, MPS methods revert to a backup policy that simply instructs halting, rather than trying to find a more optimal recovery approach, such as finding a route that steers around the obstacle. As a result, the recovery phase of MPS frequently inhibits the learning process and incurs high *recovery regret*, meaning that there is a large discrepancy between the return from executed recovery actions and the maximum possible returns from the same states.

Motivated by this problem, we propose a novel PSRL approach called *Dynamic Model Predictive Shielding* (DMPS), which aims to optimize RL objectives while ensuring provable safety under complex dynamics. The key idea behind DMPS is to employ a local planner [11, 12] to dynamically identify a safe recovery action that optimizes finite-horizon progress towards the goal. Although the computational overhead of planning grows exponentially with the depth of the horizon, in practice, a reasonably small horizon is sufficient for allowing the agent to recover from unsafe regions.

In DMPS, the planner and the neural policy under training play a synergistic role. First, when using the planner to recover from potentially unsafe regions, our optimization objective not only uses the finite-horizon reward but also uses the Q-function learned by the neural policy. The integration of the Q-function into the optimization objective allows the planner to take into account *long-term reward* beyond the short planning horizon needed for recovering safety. As a result, the planner benefits from the neural policy in finding recovery actions that optimize for long-term reward. Conversely, the integration of the planner into the training loop allows the neural policy to learn from actions suggested by the planner: Because the planner dynamically figures out how to avoid unsafe regions while making task-specific progress, the policy under training also learns how to avoid unsafe regions in a *smart* way, rather than taking overly conservative actions. From a theoretical perspective, DMPS can guarantee provable safety both during and after training. We also provide a theoretical guarantee that the regret from recovery trajectories identified by DMPS is bounded and that it decays at an exponential rate with respect to the depth of the planning horizon. These properties allow DMPS to learn policies that are both high-performing and safe.

We have implemented the DMPS algorithm in an open-source library and evaluated it on a suite of 13 representative benchmarks. We experimentally compare our approach against Constrained Policy Optimization (CPO) [3], PPO-Lagrangian (PPO-Lag) [13, 14], Twin Delayed Deep Deterministic Policy Gradient (TD3) [15] and state-of-the-art PSRL approaches, MPS [9] and REVEL [16]. Our results indicate that policies learned by DMPS outperform all baselines in terms of total episodic return and achieve safety with minimal shield invocations. Specifically, DMPS invokes the shield 76% less frequently compared to the next best baseline, MPS, and achieves 29% higher returns after convergence.

To summarize, the contributions of this paper are as follows: First, we introduce the DMPS algorithm, a novel integration of RL and planning, that aims to address the limitations of model predictive shielding. Second, we provide a theoretical analysis of our algorithm and prove that recovery regret decreases exponentially with planning horizon depth. Lastly, we present a detailed empirical evaluation of our approach on a suite of PSRL benchmarks, demonstrating superior performance compared to several state-of-the-art baselines.

2 Related Work

PSRL. There is a growing body of work addressing safety issues in RL [1, 2, 17, 18, 19, 20, 21]. Our approach falls into the category of provably safe RL (PSRL) techniques [7, 18], treating safety as a hard constraint that must never be violated. This is in contrast to *statistically safe* RL techniques, which provide only statistical bounds on the system's safety by constraining the training objectives [3, 22, 23, 24, 25]. These soft guarantees, however, are insufficient for domains like autonomous driving, where each failure can be catastrophic. Existing PSRL methods can be categorized based on whether

they guarantee safety *throughout the training phase* [26, 27, 16, 28] or only *post-training* upon deployment [29, 30, 31, 32, 33]. Our approach falls into the former category and employs a shielding mechanism that ensures safety both during training and deployment.

Safety Shielding. Many PSRL works use *safety shields* [8, 34, 35, 36, 37, 16, 38, 39]. These methods typically synthesize a domain-specific policy ahead of time and use it to detect and substitute unsafe actions at runtime. However, traditional shielding methods tend to be computationally intractable, leading to limited usability. For instance, [8] presents one of the early works in shielding, where safety constraints are specified in Linear Temporal Logic (LTL), and a verified reactive controller is synthesized to act as the safety shield at runtime. However, this approach is restricted to discrete state and action spaces, due to the complexity of shield synthesis. Another example is [39], which enhances PSRL performance by substituting shield-triggering actions with verified actions and reducing the frequency of shield invocations. In Model Predictive Shielding (MPS) [9], a backup policy dynamically assesses the states from which safety can be reinstated and, if necessary, proposes substitute actions. Unlike pre-computed shields, MPS can handle high-dimensional, non-linear systems with continuous states and actions, without incurring an exponential runtime overhead [40]. MPS has also been successfully applied to stochastic dynamics systems [41] and multi-agent environments [10]. However, a significant limitation of current MPS methods is the separation of safety considerations from the RL objectives, which hinders learning when employing the recovery policy.

RL and Planning. Planning has traditionally been considered a viable complement to reinforcement learning, combining real-time operations in both the actual world and the learned environment model [42, 43, 44, 45, 46]. With recent developments in deep model-based RL [44, 47, 48, 49, 50] and the success of planning algorithms in discrete [51, 52] and continuous spaces [53, 54, 55, 56, 57], the prospect of combining these methods holds great promise for solving challenging real-world problems. Integrating planning within RL has also been applied to safety measures [58, 59, 21, 60, 61, 62, 63, 64]. For instance, Safe Model-Based Policy Optimization [21] minimizes safety violations by detecting non-recoverable states through forward planning using an accurate dynamics model of the system. However, it only employs planning to identify unsafe states, not to find optimal recovery paths from such situations. To the best of our knowledge, `DMPS` is the first method that leverages dynamic planning for optimal recovery, within the framework of model predictive shielding.

Classical Control. There is a long line of classical control approaches to the problem of safe navigation [65, 66, 67, 68, 69, 70]. One common approach, for instance, is the use of *control barrier functions* (CBFs) [71, 72]. CBFs have been used across many domains in robotics to achieve safety with high confidence [73, 74, 75, 76]. While useful, these approaches tend to make assumptions about the environment (e.g. differentiability, closed-form access to dynamics) that cannot easily be reconciled with the highly general RL framing of this work.

3 Preliminaries

MDP. We formulate our problem using a standard Markov Decision Process (MDP) $\mathcal{M} = \langle \mathcal{S}, \mathcal{S}_U, \mathcal{S}_0, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma \rangle$, where $\mathcal{S} \subseteq \mathbb{R}^n$ is a set of states, $\mathcal{S}_U \subset \mathcal{S}$ is a set of unsafe states, $\mathcal{S}_0 \subset \mathcal{S}$ are the initial states, $\mathcal{A} \subseteq \mathbb{R}^m$ is a continuous action space, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is a deterministic transition function, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a deterministic reward function, and γ is the discount factor. We define Π to be the set of all agent policies, where each policy $\pi \in \Pi$ is a function from environment states to actions, *i.e.,* $\pi : \mathcal{S} \to \mathcal{A}$. For any set $S \subseteq \mathcal{S}$, the set of *reachable states* from S in i steps under policy π, is denoted by $\texttt{reach}_i(\pi, S)$, and is defined recursively as follows: $\texttt{reach}_1(\pi, S) \doteq \{\mathcal{T}(s, \pi(s)) \mid s \in S\}$ and $\texttt{reach}_{i+1}(\pi, S) \doteq \texttt{reach}_1(\pi, \texttt{reach}_i(\pi, S))$. The set of *all* reachable states under a policy π is $\texttt{reach}(\pi) \doteq \bigcup_{1 \leq i} \texttt{reach}_i(\pi, \mathcal{S}_0)$.

PSRL. The standard objective in RL is to find a policy π that maximizes a performance measure, $J(\pi)$, typically defined as the expected infinite-horizon discounted total return. In provably safe reinforcement learning (PSRL), the aim is to identify a *safe* policy that maximizes the above measure. The set of safe policies is denoted by $\Pi_{\textit{safe}} \subseteq \Pi$ and consists of policies that never reach an unsafe state, *i.e.,* $\pi \in \Pi_{\textit{safe}} \Leftrightarrow \texttt{reach}(\pi) \cap \mathcal{S}_U = \emptyset$. Therefore, the objective of PSRL is to find a policy $\pi^*_{\textit{safe}} \doteq \arg\max_{\pi \in \Pi_{\textit{safe}}} J(\pi)$.

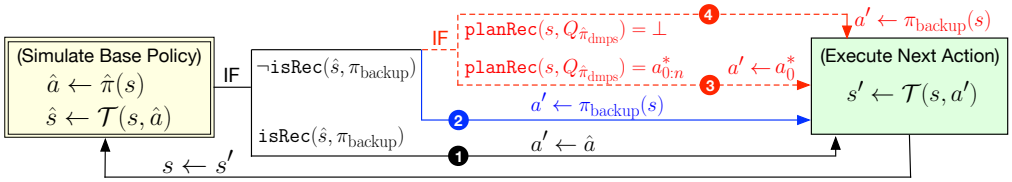


Figure 1: Overview of an execution cycle in MPS (❶, ❷) and DMPS (❶, ❷, ❸, ❹).

4 Model Predictive Shielding

Under the Model Predictive Shielding (MPS) framework [9], a safe policy, π_{mps}, is constructed by integrating two sub-policies: a *learned policy*, $\hat{\pi}_{\text{mps}}$, and a *backup policy*, π_{backup}. Depending on the current state of the system, control of the agent's actions is delegated to one of these two policies. The learned policy is typically implemented as a neural network that is trained using standard deep RL techniques to optimize $J(\cdot)$. However, this policy may violate safety during training or deployment, *i.e.,* $\hat{\pi}_{\text{mps}} \notin \Pi_{\text{safe}}$. On the other hand, the backup policy, π_{backup}, is specifically designed for safety and is invoked to substitute potentially unsafe actions proposed by the learned policy.

Due to domain-specific constraints on system transitions, the backup policy is effective only from a certain subset of states, called *recoverable* states. For instance, given the deceleration limits of an agent, a backup policy that instructs the agent to halt can only avoid collisions from states where there is sufficient distance between the agent and the obstacle. At a high level, the recoverability of a given state s in MPS is determined by a function $\texttt{isRec} : \mathcal{S} \times \Pi \rightarrow \mathbb{B}$. This function performs an N-step forward simulation of π_{backup} from s and checks whether a safety equilibrium can be established.

Figure 1 gives an overview of the control delegation mechanism in π_{mps}.[1] Given state $s \in \mathcal{S}$, the agent first forecasts the next state $\hat{s}$ that would be reached by following the learned policy (double-bordered, yellow box). If $\texttt{isRec}(\hat{s}, \pi_{\text{backup}})$ is true, then π_{mps} simply returns the same action as $\hat{\pi}_{\text{mps}}$, as marked by ❶. Otherwise, if $\texttt{isRec}(\hat{s}, \pi_{\text{backup}})$ is false, π_{mps} delegates control to the backup policy π_{backup}, as indicated by ❷. The selected action, a', is then executed in the environment (single-bordered, green box), resulting in a new state s'. From this state, π_{mps} is executed again, and the process repeats.

The safety of π_{mps} relies on the fact that all recoverable states are safe and that the backup policy is *closed* under the set of recoverable states. Thus, we can inductively show that the agent remains in safe and recoverable states; thus $\pi_{\text{mps}} \in \Pi_{\textit{safe}}$.

Example.

Consider an agent on a 2D plane aiming to reach a goal while avoiding static obstacles. Figure 2 (a) presents an unsafe trajectory proposed by the learned policy. Figure 2 (b) presents the trajectory under MPS. As discussed above, π_{mps} delegates control to the backup policy in such potentially unsafe situations, which corresponds here to applying maximum deceleration away from the obstacle. This causes the agent to come to a halt, which is suboptimal. Figure 2 (c) depicts a DMPS trajectory. Figure 2 (d) depicts DMPS planning in an environment containing a static obstacle and a low-reward puddle region. The latter two are explained in section 5.

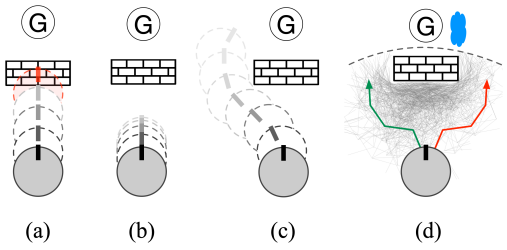


Figure 2: (a) Unsafe trajectory leading to a collision. (b) Safe but sub-optimal trajectory. (c) Optimal and safe trajectory. (d) An instance of the planning phase.

4.1 Recovery Regret

While MPS guarantees worst-case safety, shield interventions can hinder the training of $\hat{\pi}_{\text{mps}}$ and compromise overall efficacy. To formalize this limitation, we introduce the concept of *Recovery*

[1] Arrows ❸ and ❹ are used for planner-guided recovery, and are explained more in section 5.

Regret, which measures the expected performance gap between π_{backup} and the optimal policy. To this end, we first introduce a helper function $\mathbb{I}_{\text{backup}} : \mathcal{S} \to \{0, 1\}$, which, given a state s, indicates whether control in s is delegated to the backup policy, *i.e.,* $\mathbb{I}_{\text{backup}}(s) = 1$ if $\neg\texttt{isRec}(\mathcal{T}(s, \hat{\pi}_{\text{mps}}(s), \pi_{\text{backup}}))$ and $\mathbb{I}_{\text{backup}}(s) = 0$ otherwise. In any state s where control is delegated to the backup policy, the *optimal value* $V^*(s)$ represents the maximum expected reward that can be achieved from s, and the *optimal action-value* $Q^*(s, \pi_{\text{backup}}(s))$ represents the value of executing π_{backup} in s and thereafter following an optimal policy. Thus, we can define Recovery Regret (RR) as the expected discrepancy between these two values across states where π_{backup} is invoked, *i.e.,*

$$RR(\pi_{\text{backup}}, \pi_{\text{mps}}) \doteq \mathbb{E}_{s\sim\rho^{\pi_{\text{mps}}}}\Big[\mathbb{I}_{\text{backup}}(s) \cdot \Big[V^*(s) - Q^*(s, \pi_{\text{backup}}(s))\Big]\Big]$$

In the above formula, $\rho^{\pi_{\text{mps}}}$ is the *discounted state visitation distribution* associated with π_{mps}, *i.e.,* $\rho^{\pi_{\text{mps}}}(s) \doteq (1-\gamma)\sum_{t=0}^{\infty}\gamma^t Pr(s_t = s \mid s_0 \in \mathcal{S}_0,\ \forall_{i\geq 0}.\ s_{i+1} = \mathcal{T}(s_i, \pi_{\text{mps}}(s_i)))$. This term assists in quantifying how frequently each state is visited when measuring the regret of a backup policy.

In existing MPS approaches, the misalignment between backup policies and the optimal policy can result in substantial recovery regrets. For example, Figure 2 (c) illustrates the optimal sequence of actions in the previously discussed scenario, where the agent avoids a collision by maneuvering around the obstacle. However, in MPS, the training process lacks mechanisms to teach such optimal behavior to $\hat{\pi}_{\text{mps}}$ and only teaches overly cautious and poorly rewarded actions depicted in Figure 2 (b).

5 Dynamic Model Predictive Shielding

In this section, we introduce the Dynamic Model Predictive Shielding (DMPS) framework that builds on top of MPS but aims to address its shortcomings. Similar to the control delegation mechanism in MPS, the policy π_{dmps} initially attempts to select its next action using a learned policy $\hat{\pi}_{\text{dmps}}$. If the action proposed by $\hat{\pi}_{\text{dmps}}$ leads to a state that is not recoverable using π_{backup}, an alternative safe action is executed instead, as in standard MPS. However, rather than using the task-oblivious policy π_{backup}, the backup action is selected using a *dynamic backup policy* π^*_{backup}. More formally,

$$\pi_{\text{dmps}}(s) = \begin{cases} \hat{\pi}_{\text{dmps}}(s) & \text{if } \texttt{isRec}(\mathcal{T}(s, \hat{\pi}_{\text{dmps}}(s)), \pi_{\text{backup}}) \\ \pi^*_{\text{backup}}(s) & \text{otherwise} \end{cases} \tag{1}$$

The core innovation behind DMPS is the dynamic backup policy π^*_{backup}, which is realized using a *local planner* [45, 77], denoted as a function $\texttt{planRec}()$. At a high level, planRec performs a finite-horizon lookahead search over a predetermined number of steps ($n \in \mathbb{N}$) and identifies a sequence of backup actions optimizing the expected returns during recovery. Specifically, the function planRec takes an initial state s_0 and a state-action value function $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, and returns a sequence of task-optimal actions $a^*_{0:n} \in \mathcal{A}^{n+1}$ defined as follows:

$$\begin{aligned} \texttt{planRec}(s_0, Q) \doteq \underset{a_{0:n}\in\mathcal{A}^{n+1}}{\arg\max} \Big[(\sum_{i=0}^{n-1}\gamma^i \cdot \mathcal{R}(s_i, a_i)) + \gamma^n \cdot Q(s_n, a_n)\Big], \\ \text{such that, } \forall_{0\leq i<n}[s_{i+1} = \mathcal{T}(s_i, a_i)] \text{ and } \forall_{0\leq i\leq n}\texttt{isRec}(s_i, \pi_{\text{backup}}). \end{aligned} \tag{2}$$

Crucially, the recovery plan is required to only lead to *recoverable* states within the finite planning horizon (*i.e.,* $\texttt{isRec}(s_i, \pi_{\text{backup}})$). Note that the backup policy π_{backup} is used by the planner in deciding the recoverability of a state. Beyond satisfying the hard safety constraint, the planner is also required to optimize the objective function shown in Equation 2. Importantly, this objective accounts for both the immediate rewards within the planning horizon, $\mathcal{R}(s_i, a_i)$, *as well as* the estimated long-term value from the terminal state s_n beyond the planning horizon, as defined by Q. As a result, the planner benefits from the long-term reward estimates learned by the neural policy $\hat{\pi}_{\text{dmps}}$.

Figure 2 (d) illustrates an agent planning a recovery path around an obstacle. The actions considered by planRec are represented by gray edges. Two optimal paths on this tree, depicted in green and red, yield similar rewards due to the symmetric nature of the reward function relative to the goal position. The planner selects the green path on the left as its final choice due to a water puddle on the right side of the goal, which, if traversed, would result in lower returns. Since the puddle lies outside of the planning horizon, the decision to opt for the green path is informed by access to the

Algorithm 1: Reinforcement Learning with Dynamic Recovery Planning

Inputs: ($\mathcal{M}$: Markov Decision Process), (E: Episode Count), (π_{backup}: Task-oblivious backup Policy)
Output: ($\hat{\pi}_{\text{dmps}}$: Optimal Learned Policy)
$\hat{\pi}_{\text{dmps}} :=$ `initNeuralPolicy()` # Initialize a neural network to act as the learned policy.
$\mathcal{B} := \emptyset$ # Initialize an empty replay buffer.
for $e \in [1, E]$ **do**
  $s :=$ `beginEpisode`(e) # Initialize a new episode and receive the first observed state.
  while $\neg$`terminated`(e) **do**
    $a_{\text{next}} := \hat{\pi}_{\text{dmps}}(s)$ # Choose a candidate for the next action using the learned policy.
    if $\neg$`isRec`$(\mathcal{T}(s, a_{\text{next}}), \pi_{\text{backup}})$ **then**
      $\mathcal{B} := \mathcal{B} \cup \langle s, a_{\text{next}}, \varnothing, r^{-} \rangle$ # Record a large negative penalty for triggering the backup policy.
      if $a^*_{0:n} =$ `planRec`$(s, Q_{\hat{\pi}_{\text{dmps}}})$ **then** $a_{\text{next}} := a^*_0$ # Plan recovery and choose the next action.
      else $a_{\text{next}} := \pi_{\text{backup}}(s)$ # Use the task-oblivious backup policy if planning fails.
    $s', r :=$ `execute`(s, a_{next}) # Execute the selected next action on the environment.
    $\mathcal{B} := \mathcal{B} \cup \langle s, a_{\text{next}}, s', r \rangle$ # Update the buffer with the recent transition record.
    $s := s'$
  `updateNeuralPolicy`$(\hat{\pi}_{\text{dmps}}, \mathcal{B})$ # Update the learned policy using buffered records.
return $\hat{\pi}_{\text{dmps}}$

Q-function. By executing the first action on the green path and repeating dynamic recovery, the agent can demonstrate the desired behavior shown in Figure 2 (c).

Given the plan returned by `planRec`, the dynamic backup policy π^*_{backup} returns the first action in the plan a^*_0 as its output. However, `planRec` could, in theory, fail to find a safe and optimal plan, even though one exists. In such cases, `planRec` returns a special symbol $\perp$, and π^*_{backup} reverts to the task-oblivious backup policy π_{backup}. Thus, we have:

$$\pi^*_{\text{backup}}(s) = \begin{cases} a^*_0 & \text{If } \texttt{planRec}(s, Q_{\hat{\pi}_{\text{dmps}}}) = a^*_{0:n} \\ \pi_{\text{backup}}(s) & \text{If } \texttt{planRec}(s, Q_{\hat{\pi}_{\text{dmps}}}) = \perp \end{cases} \tag{3}$$

An outline of π_{dmps} is shown in Figure 1 where the control delegation mechanism is represented by the red dashed arrows (marked as ❸ and ❹) instead of the blue solid arrow (marked as ❷).

5.1 Planning Optimal Recovery

The specific choice of the planning algorithm to solve Equation 2 depends on the application domain. However, DMPS requires two main properties to be satisfied by the planner: *probabilistic completeness* and *asymptotic optimality*. The former property states that the planner will eventually find a solution if one exists, while the latter states that the found plan converges to the optimal solution as the allocated resources increase. There exist several state-of-the-art planners that satisfy both of these requirements, including sampling-based planners such as RRT* [78] and MCTS [79], which have been shown to be particularly effective at finding high-quality solutions in high-dimensional continuous search spaces [53, 54, 55]. These methods construct probabilistic roadmaps or search trees and deal with the exponential growth in the search space by incrementally exploring and expanding only the most promising nodes based on heuristics or random sampling. Given an implementation of `planRec` that satisfies the aforementioned requirements, a significant outcome in DMPS is that, as the depth of the planning horizon (n) increases, the expected return from π^*_{backup} approaches the globally optimal value. The following theorem states the optimality of recovery in π_{dmps}, in terms of exponentially diminishing recovery regret of π^*_{backup} as n increases.

Theorem 5.1. [2] ***(Simplified)*** *Suppose the use of a probabilistically complete and asymptotically optimal planner with planning horizon n and sampling limit m. Under mild assumptions of the MDP, the recovery regret of policy π^*_{backup} used in π_{dmps} is almost surely bounded by order γ^n as m goes to infinity. In other words, with probability 1,*

$$\lim_{m \to \infty} RR(\pi^*_{backup}, \pi_{\text{dmps}}) = \mathcal{O}(\gamma^n).$$

5.2 Training Algorithm

[2] Extended theorem statement and proof are provided in Appendix A.1.

While our proposed recovery can be used during deployment irrespective of how the neural policy is trained, our method integrates the planner into the training loop, allowing the neural policy to learn to "imitate" the safe actions of the planner while making task-specific progress. Hence, the training loop converges to a neural policy that is both high-performant *and* safe. This is very desirable because DMPS can avoid expensive shield interventions that require on-line planning during deployment.

Algorithm 1 presents a generic deep RL algorithm for training a policy $\hat{\pi}_{\text{dmps}}$. Lines 3-4 of the algorithm perform initialization of the neural policy $\hat{\pi}_{\text{dmps}}$ as well as the *replay buffer* $\mathcal{B}$, which stores a set of tuples $\langle s, a, s', r \rangle$ that capture transitions and their corresponding reward r. Each training episode begins with the agent observing the initial state s (line 6) and terminates when the goal state is reached or after a predetermined number of steps are taken (line 7). The agent first attempts to choose its next action a_{next} using the learned policy $\hat{\pi}_{\text{dmps}}$ (Line 8). If the execution of a_{next} leads to a non-recoverable state according to π_{backup} (line 9), the algorithm first adds a record to the replay buffer where the current state and action are associated with a high negative reward r^- (line 10). It then performs dynamic model predictive shielding to ensure safety (lines 11-12) as discussed earlier: If the planner yields a valid policy, the the next action is chosen as the first action in the plan (line 11); otherwise, the backup policy π_{backup} is used to determing the next action. Then, a_{next} is executed at line 13 to obtain a new state s' and its corresponding reward r. This new transition and its corresponding reward are again added to the replay buffer (line 14), which is then used to update the neural policy at line 16, after the termination of the current training episode.

6 Experiments

In this section, we present an empirical study of DMPS on 13 benchmarks and compare it against 4 baselines. The details of our implementation and experimental setup are presented in Appendix A.3.

Benchmarks. We evaluate our approach on five *static benchmarks* (ST), where the agent's environment is static, and eight *dynamic benchmarks*, where the agent's environment includes moving objects. Static benchmarks (`obstacle`, `obstacle2`, `mount-car`, `road`, and `road2d`) are drawn from prior work [16, 27] and include classic control problems. The more challenging dynamic benchmarks (`dynamic-obst`, `single-gate`, `double-gates`, and `double-gates+`) require the agent to adapt its policy to accommodate complex obstacle movements. Specifically, `dynamic-obs` features moving obstacles along the agent's path. In `single-gate`, a rotating circular wall with a small opening surrounds the goal position. `double-gate` is similar but includes *two* concentric circular walls around the goal, and `double-gate+` is the most challenging, featuring increased wall thickness to force more efficient navigation through the openings. For each dynamic benchmark, we consider two different agent dynamics: *differential drive dynamics* (DD), featuring an agent with two independently driven wheels, and *double integrator dynamics* (DI), where the agent's acceleration in any direction can be adjusted by the policy. A detailed description of benchmarks can be found in Appendix A.3.

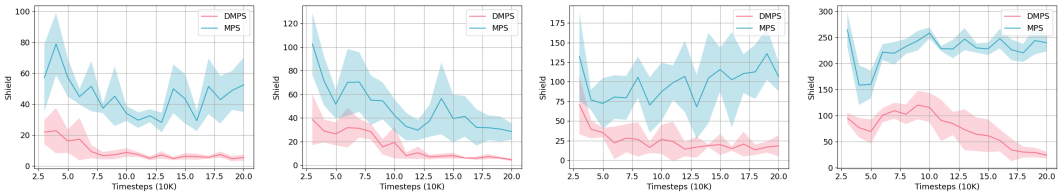
Baselines. We compare DMPS (with a planning horizon of $n = 5$) with five baselines. Our first baseline is MPS, the standard model predictive shielding approach. The second baseline is REVEL [16], a recent PSRL approach that learns verified neurosymbolic policies through iterative mirror descent. REVEL requires a white-box model of the environment's worst-case dynamics, which is challenging to develop for dynamic benchmarks; hence, we apply REVEL only to static benchmarks. We also compare DMPS against three established RL methods: CPO [3], PPO-Lag (PPO Lagrangian) [13, 14], and TD3 [15]. All aim to reduce safety violations during training by incorporating safety constraints into the objective. In TD3, a negative penalty is applied to unsafe steps. In CPO, a fixed number of violations are tolerated. Finally, in PPO-Lag, a Lagrangian multiplier is used to balance safety cost with reward.

6.1 Safety Results

Table 1 presents our experimental results regarding safety. All results are averaged over 5 random seeds. For PSRL methods (DMPS, MPS, and REVEL), which guarantee worst-case safety, we report the average number of shield invocations per episode. Generally, less frequent shield invocation indicates higher performance of the approach. For SRL methods (TD3, PPO-Lag and CPO), we report the average number of safety violations per episode.

Table 1: Safety Results

	Benchmark	# Shield Invocations / Episode						# Safety Violations / Episode					
		DMPS		MPS		REVEL		CPO		PPO-lag		TD3	
		mean	sd	mean	sd	mean	sd	mean	sd	mean	sd	mean	sd
ST	obstacle	**0.0**	0.0	**0.0**	0.0	4.0	8.0	**0.0**	0.0	**0.0**	0.0	**0.0**	0.0
	obstacle2	**0.0**	0.0	**0.0**	0.0	33.8	30.3	0.9	0.2	6.42	0.19	**0.0**	0.0
	mount-car	**0.0**	0.0	**0.0**	0.0	4.0	4.0	2.0	2.1	22.2	28.9	**0.0**	0.0
	road	**0.0**	0.0	**0.0**	0.0	0.8	0.74	**0.0**	0.0	**0.0**	0.0	**0.0**	0.0
	road2d	**0.0**	0.0	**0.0**	0.0	**0.0**	0.0	**0.0**	0.0	**0.0**	0.0	**0.0**	0.0
DI	dynamic-obst	**9.3**	1.8	144.1	26.2	-	-	1.7	0.8	3.6	3.9	**0.8**	1.2
	single-gate	**0.1**	0.0	0.2	0.1	-	-	2.0	1.2	10.0	5.5	**0.0**	0.0
	double-gates	**4.5**	1.2	28.3	6.9	-	-	2.1	1.7	11.8	6.3	**0.3**	0.5
	double-gates+	**24.2**	6.4	239.8	16.3	-	-	2.9	1.0	6.5	4.5	**0.0**	0.0
DD	dynamic-obst	**105.2**	39.9	144.9	39.9	-	-	1.7	1.9	2.9	2.2	**0.8**	0.16
	single-gate	**3.1**	1.6	7.4	6.9	-	-	5.2	3.5	6.7	7.2	**0.1**	0.2
	double-gates	**5.5**	1.9	52.5	17.6	-	-	3.9	1.7	7.9	9.2	**3.0**	5.5
	double-gates+	**18.4**	13.0	106.2	18.5	-	-	12.1	2.9	6.9	6.1	**0.2**	0.4



double-gate (DD) double-gate (DI) double-gate+ (DD) double-gate+ (DI)

Figure 3: Shield Invocations in double-gate and double-gate+

Due to the relative simplicity of the static benchmarks, the PSRL baselines achieve safety with very infrequent shield invocations. Even the SRL baselines are mostly able to avoid safety violations in these benchmarks. On the other hand, in dynamic benchmarks, PSRL approaches heavily rely on shielding to achieve safety, and SRL approaches violate safety more frequently. Notably, the number of DMPS shield invocations is significantly lower than in other baselines. Across all dynamic benchmarks, DMPS invokes the shield an average of 24.7 times per episode, whereas MPS triggers the shield 124.1 times. The results also indicate that the standard deviation values for shield invocations in DMPS are consistently lower than those of MPS, indicating more stable and predictable performance.

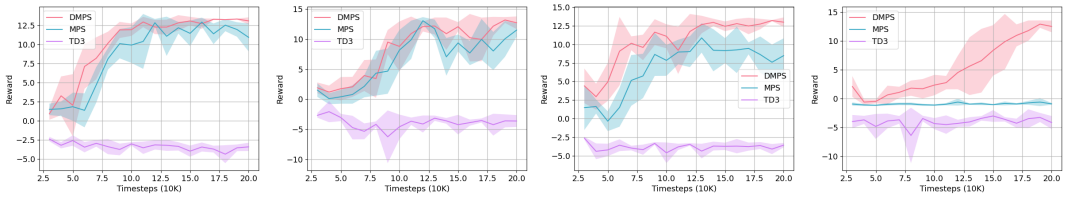
Figure 3 shows the number of shield triggers plotted against episodes for the double-gate and double-gate+ dynamic benchmarks. The error regions are 1-sigma over random seeds. Under both agent dynamics, DMPS achieves significantly fewer shield invocations compared to MPS. Moreover, the number of shield invocations in DMPS consistently decreases as training progresses, whereas this trend is not present in MPS. In fact, in many scenarios, the number of shield invocations in MPS increases with more training because the learned policy reduces its exploratory actions and adheres more strictly to a direct path to the goal. However, this frequently leads to being blocked by obstacles and results in repeated shield invocations.

6.2 Performance Results

Table 2 presents the per-episode return over the last 10 test episodes of a run. The reported mean and standard deviations are computed over 5 random seeds. The results indicate that DMPS and MPS exhibit comparable performance across most static benchmarks, with the exception of the more challenging obstacle and obstacle2 benchmarks for which DMPS significantly outperforms MPS. In dynamic benchmarks, DMPS outperforms MPS in all benchmarks except for single-gate (DI), where both methods achieve equivalent results. Both DMPS and MPS consistently outperform REVEL. The SRL approaches (CPO, PPO-Lag, and TD3) perform reasonably well in most static benchmarks but their performance significantly deteriorates in dynamic benchmarks, failing to achieve positive returns in any instance. This is because the policies in CPO, PPO-Lag, and TD3 rapidly learn to avoid both the obstacles and the goal due to harsh penalties for safety violations, thus accruing the negative rewards for each timestep spent away from the goal.

Table 2: Performance Results

	Benchmark	Total Return in Final 10 Episode											
		DMPS		MPS		REVEL		CPO		PPO-Lag		TD3	
		mean	sd	mean	sd	mean	sd	mean	sd	mean	sd	mean	sd
ST	obstacle	32.7	0.3	8.6	47.9	−41.6	52.7	32.8	0.0	32.8	0.4	**32.9**	0.0
	obstacle2	20.2	15.2	−1.8	3.2	9.3	21.1	33.0	0.2	**34.1**	0.1	−1.2	3.0
	mount-car	81.2	0.3	**85.1**	1.8	11.4	34.1	9.6	35.3	−21.3	2.9	−30.0	35.5
	road	**22.7**	0.0	**22.7**	0.0	9.7	16.4	**22.7**	0.0	**22.7**	0.05	**22.7**	0.0
	road2d	**24.0**	0.2	**24.0**	0.2	11.2	16.5	**24.0**	0.0	**24.0**	0.1	**24.0**	0.1
DI	dynamic-obs	**13.2**	0.0	−1.3	1.9	-	-	−21.4	21.1	−4.2	24.9	−5.0	0.3
	single-gate	**11.6**	0.0	**11.6**	0.0	-	-	−19.6	17.0	−2.0	1.1	−2.1	0.2
	double-gates	**12.7**	1.0	11.5	1.1	-	-	−6.7	5.4	−3.9	11.1	−3.6	1.0
	double-gates+	**13.0**	0.6	−0.9	0.1	-	-	−17.4	12.8	−2.8	0.4	−4.1	1.1
DD	dynamic-obst	**7.4**	3.7	6.3	2.1	-	-	−4.5	0.5	−3.6	0.8	−5.3	0.5
	single-gate	**11.5**	0.1	11.4	0.1	-	-	−2.5	0.3	−2.4	0.2	−3.1	0.5
	double-gates	**13.1**	0.5	10.9	1.8	-	-	−2.4	0.6	−1.9	0.9	−3.4	0.5
	double-gates+	**13.0**	0.6	8.5	2.3	-	-	−2.5	0.3	−20.7	22.8	−3.6	0.3



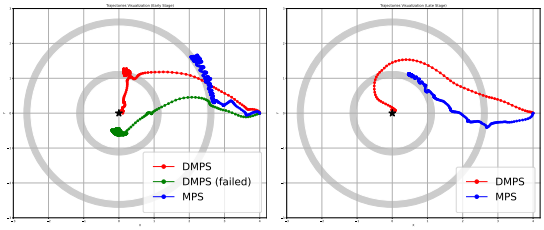
double-gate (DD) double-gate (DI) double-gate+ (DD) double-gate+ (DI)

Figure 4: Episodic Returns in double-gate and double-gate+

Figure 4 presents the total return plotted against the episode number for the double-gate and double-gate+ dynamic benchmarks. The error regions are 1-sigma over random seeds. The CPO and PPO-Lag baselines are omitted due to their significantly poor performance, which distorts the scale of the graphs. In all benchmarks, DMPS demonstrates superior performance compared to the other methods. While MPS performs adequately in three of the benchmarks, it exhibits poor performance in the double-gate+ (DI) benchmark.

6.3 Analysis

We analyze the performance of DMPS and MPS on the double-gate+ environment under double-integrator dynamics. Figure 5a shows trajectories from the first half of training when the agent's policy and Q function are still under-trained. When the agents approach the first gate and attempt to cross it, the shield is triggered. In the case of MPS, this shield simply pushes the agent back outside the gate (see blue trajectory), and the agent is unable to make any progress. In contrast, DMPS plans actions that maximize an n-step return target, allowing the agent to initially make progress. However, we can observe from the green trajectory that the agent's Q function and policy are under-trained: after having made it through the obstacles, the agent is not sufficiently goal-aware to reliably make it to the center. This makes the DMPS planner objective inaccurate with respect to long-term return, causing suboptimal actions to be selected by the shield. In the red trajectory, for instance, the agent spends a large portion of the trajectory trying to cross the second gate, only to retreat and get stuck in that position until eventually getting through.



(a) Early Stage (b) Late Stage

Figure 5: Example trajectories in double-gate+ .

As training progresses, the MPS backup policy hinders the agent's exploration of the environment, impeding the learning of the neural policy. By the end, most runs of the MPS agent are unable to progress past the first gate. Figure 5b shows one of the few MPS runs that successfully make it through the first gate towards the end of training; however, the MPS agent still fails to make it through

the second gate. In contrast, the DMPS agent can learn from the actions suggested by the planner, improving its policy and Q function. This improvement makes the DMPS planner's objective a more accurate estimation of long-term return, further strengthening the planner. Consequently, the DMPS agent demonstrates significantly better behaviors, as shown by the red trajectory in Figure 5b.

7 Limitations

7.1 Determinism

First, our approach requires a perfect-information deterministic model, which could limit its usability in real-world deployment. Much prior work on provably safe locomotion makes the same determinism assumptions that we do [10, 31, 80], with some such algorithms even having been deployed on real robots [3]. However, extending our DMPS approach to stochastic environments is a promising direction for future work. In particular, since prior MPS work has been extended to work in stochastic settings [40], we believe our DMPS approach can be similarly extended to the stochastic setting.

7.2 Computational Overhead

Another potential limitation of our approach is the computational overhead of the planner. We use MCTS in an anytime planning setting, where we devote a fixed number of node expansions to searching for a plan. The clock time needed is linear in the allocated compute budget. However, the worst-case computational complexity to densely explore the search space, as in general planning problems, would be $O(\exp(H))$ since the planning search space grows exponentially. Our implementation used MCTS with 100 node expansions, a plan horizon of 5, and a branch factor of 10, which amounts to exploring 1000 states in total. On average, we found that when the shield is triggered, planning takes 0.4 seconds per timestep. The code we used is unoptimized, written in Python, and single-threaded. Since MCTS is a CPU-intensive search process, switching to a language C++ would likely yield significant speed improvements, and distributing computation over multiple cores would further slash the compute time by the number of assigned cores. We perform some additional experiments on the `double-gates+` environment, outlined in subsection A.4, to see how necessary compute budget scales with planner depth, confirming a rough exponential relationship.

7.3 Sufficiency of Planning Horizon

Finally, it can be asked whether small planning horizons (in our case, we used $H = 5$) are sufficient to solve tricky planning problems. Our planner objective is designed to ensure that the planner accounts for both short-term and long-term objectives, preventing overly myopic behavior even with short horizons. However, the length of the horizon still affects how close to the globally optimal solution the result is, with a tradeoff of computational cost. To see this empirically, we conducted an experiment on the `double-gates+` environment, re-evaluating it under different planner depths and observing performance differences. Despite better initial performance, the low horizon agent converged to the same performance once the policy had fully stabilized. As part of our analysis, we also found that trivial planning ($H = 1$) does not work, reaffirming the necessity of a planner. More analysis and experimental results can be found in subsection A.5.

8 Conclusions

In this paper, we proposed Dynamic Model Predictive Shielding (DMPS), which is a variant of Model Predictive Shielding (MPS) that performs dynamic recovery by leveraging a local planner. The proposed approach takes less conservative actions compared to MPS while still guaranteeing safety, and it learns neural policies that are both effective and safe in practice. Our evaluation on several challenging tasks shows that DMPS improves upon the state-of-the-art methods in Safe Reinforcement Learning.

Impact Statement. Over the past decade, reinforcement learning has experienced significant advancements and is increasingly finding applications in critical safety environments, such as autonomous driving. The stakes in such settings are high, with potential failures risking considerable property damage or loss of life. This study seeks to take a meaningful step for mitigating these risks by developing reinforcement learning agents that are rigorously aligned with real-world safety requirements.

Acknowledgements. This work is partially supported by the National Science Foundation (CCF-2319471 and CCF-1901376). Any opinions, findings, and conclusions expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

References

[1] Garcıa, J., F. Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

[2] Gu, S., L. Yang, Y. Du, et al. A review of safe reinforcement learning: Methods, theory and applications. *arXiv preprint arXiv:2205.10330*, 2022.

[3] Achiam, J., D. Held, A. Tamar, et al. Constrained policy optimization. In *International conference on machine learning*, pages 22–31. PMLR, 2017.

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

[5] Miryoosefi, S., C. Jin. A simple reward-free approach to constrained reinforcement learning. In *International Conference on Machine Learning*, pages 15666–15698. PMLR, 2022.

[6] Wagener, N. C., B. Boots, C.-A. Cheng. Safe reinforcement learning using advantage-based intervention. In *International Conference on Machine Learning*, pages 10630–10640. PMLR, 2021.

[7] Krasowski, H., J. Thumm, M. Müller, et al. Provably safe reinforcement learning: Conceptual analysis, survey, and benchmarking. *Transactions on Machine Learning Research*, 2023.

[8] Alshiekh, M., R. Bloem, R. Ehlers, et al. Safe reinforcement learning via shielding. In *Proceedings of the AAAI conference on artificial intelligence*, vol. 32. 2018.

[9] Bastani, O. Safe reinforcement learning with nonlinear dynamics via model predictive shielding. In *2021 American control conference (ACC)*, pages 3488–3494. IEEE, 2021.

[10] Zhang, W., O. Bastani, V. Kumar. Mamps: Safe multi-agent reinforcement learning via model predictive shielding. *arXiv preprint arXiv:1910.12639*, 2019.

[11] Moerland, T. M., J. Broekens, C. M. Jonker. A framework for reinforcement learning and planning. *arXiv preprint arXiv:2006.15009*, 127, 2020.

[12] Den Hengst, F., V. François-Lavet, M. Hoogendoorn, et al. Planning for potential: efficient safe reinforcement learning. *Machine Learning*, 111(6):2255–2274, 2022.

[13] Schulman, J., F. Wolski, P. Dhariwal, et al. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

[14] Altman, E. Constrained markov decision processes with total cost criteria: Lagrangian approach and dual linear program. *Math. Methods Oper. Res.*, 48(3):387–417, 1998.

[15] Fujimoto, S., H. Hoof, D. Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

[16] Anderson, G., A. Verma, I. Dillig, et al. Neurosymbolic reinforcement learning with formally verified exploration. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, pages 6172–6183. 2020.

[17] Zhao, W., T. He, R. Chen, et al. State-wise safe reinforcement learning: A survey. *arXiv preprint arXiv:2302.03122*, 2023.

[18] Xiong, N., Y. Du, L. Huang. Provably safe reinforcement learning with step-wise violation constraints. *Advances in Neural Information Processing Systems*, 36, 2024.

[19] Odriozola-Olalde, H., M. Zamalloa, N. Arana-Arexolaleiba. Shielded reinforcement learning: A review of reactive methods for safe learning. In *2023 IEEE/SICE International Symposium on System Integration (SII)*, pages 1–8. IEEE, 2023.

[20] Shperberg, S. S., B. Liu. A rule-based shield: Accumulating safety rules from catastrophic action effects. In *Proceedings of The 1st Conference on Lifelong Learning Agents*. 2022.

[21] Thomas, G., Y. Luo, T. Ma. Safe reinforcement learning by imagining the near future. *Advances in Neural Information Processing Systems*, 34:13859–13869, 2021.

[22] Wen, M., U. Topcu. Constrained cross-entropy method for safe reinforcement learning. *Advances in Neural Information Processing Systems*, 31, 2018.

[23] Alur, R., O. Bastani, K. Jothimurugan, et al. Policy synthesis and reinforcement learning for discounted ltl. *arXiv preprint arXiv:2305.17115*, 2023.

[24] Liu, Z., H. Zhou, B. Chen, et al. Constrained model-based reinforcement learning with robust cross-entropy method. *arXiv preprint arXiv:2010.07968*, 2020.

[25] Satija, H., P. Amortila, J. Pineau. Constrained markov decision processes via backward value functions. In *International Conference on Machine Learning*, pages 8502–8511. PMLR, 2020.

[26] Bacci, E., M. Giacobbe, D. Parker. Verifying reinforcement learning up to infinity. In *Proceedings of the International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, 2021.

[27] Fulton, N., A. Platzer. Safe reinforcement learning via formal methods: Toward safe control through proof and learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32. 2018.

[28] Anderson, G., S. Chaudhuri, I. Dillig. Guiding safe exploration with weakest preconditions. In *The Eleventh International Conference on Learning Representations*. 2023.

[29] Bastani, O., Y. Pu, A. Solar-Lezama. Verifiable reinforcement learning via policy extraction. *Advances in neural information processing systems*, 31, 2018.

[30] Schmidt, L. M., G. Kontes, A. Plinge, et al. Can you trust your autonomous car? interpretable and verifiably safe reinforcement learning. In *2021 IEEE Intelligent Vehicles Symposium (IV)*, pages 171–178. IEEE, 2021.

[31] Zhu, H., Z. Xiong, S. Magill, et al. An inductive synthesis framework for verifiable reinforcement learning. In *Proceedings of the 40th ACM SIGPLAN conference on programming language design and implementation*, pages 686–701. 2019.

[32] Verma, A., V. Murali, R. Singh, et al. Programmatically interpretable reinforcement learning. In *International Conference on Machine Learning*, pages 5045–5054. PMLR, 2018.

[33] Ivanov, R., J. Weimer, R. Alur, et al. Verisig: verifying safety properties of hybrid systems with neural network controllers. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, pages 169–178. 2019.

[34] Carr, S., N. Jansen, S. Junges, et al. Safe reinforcement learning via shielding under partial observability. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, pages 14748–14756. 2023.

[35] Könighofer, B., J. Rudolf, A. Palmisano, et al. Online shielding for reinforcement learning. *Innovations in Systems and Software Engineering*, 19(4):379–394, 2023.

[36] Jansen, N., B. Könighofer, S. Junges, et al. Shielded decision-making in mdps. *arXiv preprint arXiv:1807.06096*, 2018.

[37] Könighofer, B., F. Lorber, N. Jansen, et al. Shield synthesis for reinforcement learning. In T. Margaria, B. Steffen, eds., *Leveraging Applications of Formal Methods, Verification and Validation: Verification Principles*, pages 290–306. Springer International Publishing, Cham, 2020.

[38] Yang, W.-C., G. Marra, G. Rens, et al. Safe reinforcement learning via probabilistic logic shields. *arXiv preprint arXiv:2303.03226*, 2023.

[39] Thumm, J., G. Pelat, M. Althoff. Reducing safety interventions in provably safe reinforcement learning. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7515–7522. IEEE, 2023.

[40] Bastani, O., S. Li, A. Xu. Safe reinforcement learning via statistical model predictive shielding. In *Robotics: Science and Systems*, pages 1–13. 2021.

[41] Li, S., O. Bastani. Robust model predictive shielding for safe reinforcement learning with stochastic dynamics. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7166–7172. IEEE, 2020.

[42] Sutton, R. S. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In B. Porter, R. Mooney, eds., *Machine Learning Proceedings 1990*, pages 216–224. Morgan Kaufmann, San Francisco (CA), 1990.

[43] Lin, L.-J. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine learning*, 8:293–321, 1992.

[44] Moerland, T. M., J. Broekens, A. Plaat, et al. Model-based reinforcement learning: A survey. *Foundations and Trends® in Machine Learning*, 16(1):1–118, 2023.

[45] Schrittwieser, J., T. Hubert, A. Mandhane, et al. Online and offline reinforcement learning by planning with a learned model. *Advances in Neural Information Processing Systems*, 34:27580–27591, 2021.

[46] Srouji, M., H. Thomas, Y.-H. H. Tsai, et al. Safer: Safe collision avoidance using focused and efficient trajectory search with reinforcement learning. In *2023 IEEE 19th International Conference on Automation Science and Engineering (CASE)*, pages 1–8. IEEE, 2023.

[47] Efroni, Y., N. Merlis, M. Ghavamzadeh, et al. Tight regret bounds for model-based reinforcement learning with greedy policies. *Advances in Neural Information Processing Systems*, 32, 2019.

[48] Hamrick, J. B., A. L. Friesen, F. Behbahani, et al. On the role of planning in model-based deep reinforcement learning. *arXiv preprint arXiv:2011.04021*, 2020.

[49] Zhao, M., Z. Liu, S. Luan, et al. A consciousness-inspired planning agent for model-based reinforcement learning. *Advances in neural information processing systems*, 34:1569–1581, 2021.

[50] Wang, T., J. Ba. Exploring model-based planning with policy networks. In *International Conference on Learning Representations*. 2020.

[51] Silver, D., A. Huang, C. J. Maddison, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[52] Schrittwieser, J., I. Antonoglou, T. Hubert, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

[53] Hubert, T., J. Schrittwieser, I. Antonoglou, et al. Learning and planning in complex action spaces. In *International Conference on Machine Learning*, pages 4476–4486. PMLR, 2021.

[54] Kujanpää, K., A. Babadi, Y. Zhao, et al. Continuous monte carlo graph search. *arXiv preprint arXiv:2210.01426*, 2022.

[55] Kim, B., K. Lee, S. Lim, et al. Monte carlo tree search in continuous spaces using voronoi optimistic optimization with regret bounds. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, pages 9916–9924. 2020.

[56] Yee, T., V. Lisỳ, M. H. Bowling, et al. Monte carlo tree search in continuous action spaces with execution uncertainty. In *IJCAI*, pages 690–697. 2016.

[57] Rajamäki, J., P. Hämäläinen. Continuous control monte carlo tree search informed by multiple experts. *IEEE transactions on visualization and computer graphics*, 25(8):2540–2553, 2018.

[58] Leurent, E. *Safe and Efficient Reinforcement Learning for Behavioural Planning in Autonomous Driving*. Theses, Université de Lille, 2020.

[59] Huang, W., J. Ji, B. Zhang, et al. Safe dreamerv3: Safe reinforcement learning with world models. *arXiv preprint arXiv:2307.07176*, 2023.

[60] Rong, J., N. Luan. Safe reinforcement learning with policy-guided planning for autonomous driving. In *2020 IEEE International Conference on Mechatronics and Automation (ICMA)*, pages 320–326. 2020.

[61] Wang, H., J. Qin, Z. Kan. Shielded planning guided data-efficient and safe reinforcement learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2024.

[62] Leung, K., E. Schmerling, M. Zhang, et al. On infusing reachability-based safety assurance within planning frameworks for human–robot vehicle interactions. *The International Journal of Robotics Research*, 39(10-11):1326–1345, 2020.

[63] Selim, M., A. Alanwar, S. Kousik, et al. Safe reinforcement learning using black-box reachability analysis. *IEEE Robotics and Automation Letters*, 7(4):10665–10672, 2022.

[64] Rong, J., N. Luan. Safe reinforcement learning with policy-guided planning for autonomous driving. In *2020 IEEE International Conference on Mechatronics and Automation (ICMA)*, pages 320–326. IEEE, 2020.

[65] Breeden, J., D. Panagou. Predictive control barrier functions for online safety critical control. In *61st IEEE Conference on Decision and Control, CDC 2022, Cancun, Mexico, December 6-9, 2022*, pages 924–931. IEEE, 2022.

[66] Wabersich, K. P., M. N. Zeilinger. Predictive control barrier functions: Enhanced safety mechanisms for learning-based control. *IEEE Trans. Autom. Control.*, 68(5):2638–2651, 2023.

[67] Wabersich, K. P., A. J. Taylor, J. J. Choi, et al. Data-driven safety filters: Hamilton-jacobi reachability, control barrier functions, and predictive methods for uncertain systems. *IEEE Control Systems Magazine*, 43(5):137–177, 2023.

[68] Qin, Z., K. Zhang, Y. Chen, et al. Learning safe multi-agent control with decentralized neural barrier certificates. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[69] Qin, Z., D. Sun, C. Fan. Sablas: Learning safe control for black-box dynamical systems. *IEEE Robotics Autom. Lett.*, 7(2):1928–1935, 2022.

[70] Bui, M., M. Lu, R. Hojabr, et al. Real-time hamilton-jacobi reachability analysis of autonomous system with an FPGA. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2021, Prague, Czech Republic, September 27 - Oct. 1, 2021*, pages 1666–1673. IEEE, 2021.

[71] Ames, A. D., J. W. Grizzle, P. Tabuada. Control barrier function based quadratic programs with application to adaptive cruise control. In *53rd IEEE Conference on Decision and Control, CDC 2014, Los Angeles, CA, USA, December 15-17, 2014*, pages 6271–6278. IEEE, 2014.

[72] Ames, A. D., X. Xu, J. W. Grizzle, et al. Control barrier function based quadratic programs for safety critical systems. *IEEE Trans. Autom. Control.*, 62(8):3861–3876, 2017.

[73] Xu, X., P. Tabuada, J. W. Grizzle, et al. Robustness of control barrier functions for safety critical control. *CoRR*, abs/1612.01554, 2016.

[74] Xu, X., T. Waters, D. Pickem, et al. Realizing simultaneous lane keeping and adaptive speed regulation on accessible mobile robot testbeds. In *IEEE Conference on Control Technology and Applications, CCTA 2017, Mauna Lani Resort, HI, USA, August 27-30, 2017*, pages 1769–1775. IEEE, 2017.

[75] Borrmann, U., L. Wang, A. D. Ames, et al. Control barrier certificates for safe swarm behavior. In M. Egerstedt, Y. Wardi, eds., *5th IFAC Conference on Analysis and Design of Hybrid Systems, ADHS 2015, Atlanta, GA, USA, October 14-16, 2015*, vol. 48 of *IFAC-PapersOnLine*, pages 68–73. Elsevier, 2015.

[76] Wu, G., K. Sreenath. Safety-critical control of a planar quadrotor. In *2016 American Control Conference, ACC 2016, Boston, MA, USA, July 6-8, 2016*, pages 2252–2258. IEEE, 2016.

[77] Walsh, T., S. Goschin, M. Littman. Integrating sample-based planning and model-based reinforcement learning. In *Proceedings of the aaai conference on artificial intelligence*, vol. 24, pages 612–617. 2010.

[78] Karaman, S., E. Frazzoli. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894, 2011.

[79] Coulom, R. Efficient selectivity and backup operators in monte-carlo tree search. In *International conference on computers and games*, pages 72–83. Springer, 2006.

[80] Vinod, A. P., S. Safaoui, A. Chakrabarty, et al. Safe multi-agent motion planning via filtered reinforcement learning. In *2022 International Conference on Robotics and Automation, ICRA 2022, Philadelphia, PA, USA, May 23-27, 2022*, pages 7270–7276. IEEE, 2022.

[81] Silver, D., T. Hubert, J. Schrittwieser, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[82] Couëtoux, A., J.-B. Hoock, N. Sokolovska, et al. Continuous upper confidence trees. In *Learning and Intelligent Optimization: 5th International Conference, LION 5, Rome, Italy, January 17-21, 2011. Selected Papers 5*, pages 433–445. Springer, 2011.

[83] Auer, P. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov):397–422, 2002.

[84] Brockman, G., V. Cheung, L. Pettersson, et al. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[85] Achiam, J., A. Ray, D. Amodei. Safety gym. https://openai.com/research/safety-gym, 2019. Accessed: Nov 27, 2023.

A Supplemental Materials

In this section, we provide supplemental material complementary to the main content of the paper. This includes a complete proof for Theorem 5.1 and additional details omitted from section 6 due to space constraints.

A.1 Proof of Theorem 5.1

Throughout this section, we denote $Q, Q^\star$, and $V^\star$ to be the agent Q function, the optimal Q function, and the optimal V function respectively.

Our planner is parameterized by a fixed plan depth n and a sampling limit m. The planner is tasked with finding actions $a_{0:n}$ to optimize the objective $J_Q(s_0, a_{0:n}) = \left(\sum_{i=0}^{n-1} \gamma^i R_i\right) + \gamma^n Q(s_n, a_n)$. Let ϵ_m be the supremum of $J_Q(s, a^\star_{0:n}) - J_Q(s, a_{0:n})$ over $s \in \mathcal{S}$, where $a_{0:n} = \texttt{planRec}(s)$ and $a^\star_{0:n}$ is the optimal sequence of actions with respect to the J_Q objective.

We assume that our planner is probabilistically complete and asymptotically optimal. In context, this means that $\lim_{m\to\infty} \epsilon_m = 0$ with probability 1. With this grounding established, we state the extended version of Theorem 5.1.

Theorem A.1. *With probability 1, we have:*

1. *Under the assumption that Q is $\epsilon-$close to $Q^\star$, we have $\lim_{m\to\infty} RR(\pi^\star_{backup}, \pi_{dmps}) = \mathcal{O}(\epsilon\gamma^n)$.*
2. *Under the assumptions that the maximum reward per timestep is bounded, that $\mathcal{S} \times \mathcal{A}$ is compact, and that Q is a continuous function, we have $\lim_{m\to\infty} RR(\pi^\star_{backup}, \pi_{dmps}) = \mathcal{O}(\gamma^n)$.*
3. *Under the assumptions that Q and $Q^\star$ are both Lipschitz continuous, that $\|\mathcal{T}(s,a) - s\|_2$ is bounded over all $(s,a) \in \mathcal{S} \times \mathcal{A}$, and that the initial state space $\mathcal{S}_0$ and the action space $\mathcal{A}$ are both bounded, we have $\lim_{m\to\infty} RR(\pi^\star_{backup}, \pi_{dmps}) = \mathcal{O}(n\gamma^n)$.*

We first show the following general lemma.

Lemma A.2. *Fix a state $s \in \mathcal{S}$, and let $a = \pi^\star_{backup}(s)$ be the first action returned by* `planRec`*. Let $\mathcal{S}_n$ be the set of all states reachable from s in n steps, and let $B = \sup_{s'\in\mathcal{S}_n, a'\in\mathcal{A}} |Q^\star(s',a') - Q(s',a')|$. Then, $0 \le V^\star(s) - Q^\star(s,a) \le 2B\gamma^n + \epsilon_m$.*

Proof. The bound $V^\star(s) - Q^\star(s,a) \ge 0$ follows trivially from the definitions of $V^\star$ and $Q^\star$. We now prove the upper bound.

Denote $a_{0:n}$ as the sequence of actions returned by the planner. Note that $a = a_0$. Denote $a^\star_{0:n}$ as the sequence of actions that are optimal with respect to true infinite-horizon return from s (not necessarily with respect to J_Q).

We allow $R_{0:n-1}$ and $R^\star_{0:n-1}$ to be the reward sequences attained by rolling out $a_{0:n-1}$ and $a^\star_{0:n-1}$ respectively. Similarly, we allow s_n and $s^\star_n$ to be the states reached at the end of executing $a_{0:n-1}, a'_{0:n-1}$, and $a^\star_{0:n-1}$ respectively.

First, we have

$$Q^\star(s,a) \ge \left(\sum_{i=0}^{n-1} \gamma^i R_i\right) + \gamma^n Q^\star(s_n, a_n).$$

This follows immediately from the definition of $Q^\star$ and the fact that $a = a_0$.

Since $|Q^\star(s_n,a_n) - Q(s_n,a_n)| \le B$, we have

$$\left(\sum_{i=0}^{n-1} \gamma^i R_i\right) + \gamma^n Q^\star(s_n,a_n) \ge \left(\sum_{i=0}^{n-1} \gamma^i R_i\right) + \gamma^n Q(s_n,a_n) - B\gamma^n$$

$$= J_Q(s, a_{0:n}) - B\gamma^n.$$

We know that $a_{0:n}$ optimizes J_Q to within tolerance of ϵ_m, so we can write $J_Q(s, a_{0:n}) \geq J_Q(s, a^\star_{0:n}) - \epsilon_m$. We get

$$\begin{aligned} J_Q(s, a_{0:n}) - B\gamma^n &\geq J_Q(s, a^\star_{0:n}) - \epsilon_m - B\gamma^n \\ &= \left(\sum_{i=0}^{n-1} \gamma^i R^\star_i\right) + \gamma^n Q(s^\star_n, a^\star_n) - \epsilon_m - B\gamma^n \end{aligned}$$

Invoking again $|Q^\star(s^\star_n, a^\star_n) - Q(s^\star_n, a^\star_n)| \leq B$, we get

$$\begin{aligned} \left(\sum_{i=0}^{n-1} \gamma^i R^\star_i\right) + \gamma^n Q(s^\star_n, a^\star_n) - \epsilon_m - B\gamma^n \geq \left(\sum_{i=0}^{n-1} \gamma^i R^\star_i\right) + \gamma^n Q^\star(s^\star_n, a^\star_n) - \epsilon_m - 2B\gamma^n \\ = V^\star(s) - \epsilon - 2B\gamma^n. \end{aligned}$$

Putting everything together gives us $Q^\star(s, a) \geq V^\star(s) - \epsilon - 2B\gamma^n$. Rearranging this gives us the desired claim. □

With this, we can begin proving the main theorem.

Lemma A.3. ***(Part 1 of Thm A.1)*** *Under the assumption that Q is ϵ-close to $Q^\star$, we have $lim_{m\to\infty} RR(\pi^\star_{backup}, \pi_{dmps}) = \mathcal{O}(\epsilon\gamma^n)$ with probability 1.*

Proof. In the context of Lemma A.1., we see that $B \leq \epsilon$. Thus, we conclude that for all states $s \in \mathcal{S}$, we have $V^\star(s) - Q^\star(s, \pi_{\text{dmps}}(s)) \leq 2\epsilon\gamma^n + \epsilon_m$. Since RR is simply an expectation of $V^\star(s) - Q^\star(s, \pi^\star_{\text{backup}}(s))$ over some state distribution over $\mathcal{S}$, we can conclude $RR(\pi^\star_{\text{backup}}, \pi_{\text{dmps}}) \leq 2\epsilon\gamma^n + \epsilon_m$. We know from asymptotic optimality assumption that ϵ_m decays to 0 as m goes to infinity, so we get $\lim_{m\to\infty} RR(\pi^\star_{\text{backup}}, \pi_{dmps}) = \mathcal{O}(\epsilon\gamma^n)$. □

Lemma A.4. ***(Part 2 of Thm A.1)*** *Under the assumption that the maximum reward per timestep is bounded, that $\mathcal{S} \times \mathcal{A}$ is compact, and that Q is a continuous function, we have $\lim_{m\to\infty} RR(\pi^\star_{backup}, \pi_{dmps}) = \mathcal{O}(\gamma^n)$ with probability 1.*

Proof. It suffices to show that $|Q^\star(s, a) - Q(s, a)|$ is bounded over all $(s, a) \in \mathcal{S} \times \mathcal{A}$. We can then simply apply Lemma A.3 to get the $\mathcal{O}(\gamma^n)$ bound.

It is known that a continuous function with compact domain has a maximum and minimum value. Thus, there exists some C_Q such that $|Q(s, a)| < C_Q$ for all $(s, a) \in \mathcal{S} \times \mathcal{A}$.

Pick a constant R such that the absolute value of the reward at any timestep is less than R. We see that $Q^\star(s, a)$ is less than $\sum_{t=0}^{\infty} R\gamma^t = \frac{R}{1-\gamma}$ and greater than $\sum_{t=0}^{\infty}(-R)\gamma^t = \frac{-R}{1-\gamma}$. Consequently, letting $C_{Q^\star} = \frac{R}{1-\gamma}$, we get $|Q^\star(s, a)| < C_{Q^\star}$ over all $(s, a) \in \mathcal{S} \times \mathcal{A}$.

With this, $|Q^\star(s, a) - Q(s, a)| \leq |Q^\star(s, a)| + |Q(s, a)| < C_Q + C_{Q^\star}$ over all $(s, a) \in \mathcal{S} \times \mathcal{A}$. This shows the boundedness we needed to attain the asymptotic bound. □

Lemma A.5. ***(Part 3 of Thm A.1)*** *Under the assumptions that Q and $Q^\star$ are both Lipschitz continuous, that $\|\mathcal{T}(s, a) - s\|_2$ is bounded over all $(s, a) \in \mathcal{S} \times \mathcal{A}$, and that the initial state space $\mathcal{S}_0$ and the action space $\mathcal{A}$ are both bounded, we have $\lim_{m\to\infty} RR(\pi^\star_{backup}, \pi_{dmps}) = \mathcal{O}(n\gamma^n)$ with probability 1.*

Proof. Suppose that both Q and $Q^\star$ are K-Lipschitz continuous. Let $d_{\mathcal{A}}$ be the diameter of $\mathcal{A}$. Let $d_{\mathcal{T}}$ be the supremum of $\|\mathcal{T}(s, a) - s\|_2$ over $(s, a) \in \mathcal{S} \times \mathcal{A}$. We fix some arbitrary $s \in \mathcal{S}_0$ and $a \in \mathcal{A}$. Intuitively, this will act as a "central" point upon which we can estimate values for other states. Let $D = |Q(s, a) - Q^\star(s, a)|$. Let d_0 be the radius of $\mathcal{S}_0$ when centered at s.

We first prove a subclaim. Take an arbitrary $s' \in \mathcal{S}$, and let d be the ℓ_2 distance between s and s'. Then,

$$V^\star(s') - Q^\star(s', \pi^\star_{\text{backup}}(s')) \leq 2\gamma^n \left(D + 2K\sqrt{d_{\mathcal{A}}^2 + (d + nd_{\mathcal{T}})^2} \right) + \epsilon_m.$$

The idea is to invoke Lemma A.1. To do so, we need to attain a bound on B as defined by the Lemma. Consider a state s'' that is n steps away from s'. The Triangle Inequality lets us bound the distance between s' and s'' to $d_{\mathcal{T}}$. A second application of the Triangle Inequality lets us bound the distance between s and s'' to $d + d_{\mathcal{T}}$. We can further use the boundedness of the action space to note that the maximum distance between the state-action pair (s, a) and (s'', a'') for an arbitrary $a'' \in \mathcal{A}$ is $d'' = \sqrt{d_{\mathcal{A}}^2 + (d + nd_{\mathcal{T}})^2}$. Finally, we use Lischpitz continuity to get $|Q^\star(s, a) - Q^\star(s'', a'')| \leq Kd''$ and $|Q(s, a) - Q(s'', a'')| \leq Kd''$. With this,

$$\begin{aligned} &|Q^\star(s'', a'') - Q(s'', a'')| \\ \leq &|Q^\star(s'', a'') - Q^\star(s, a)| + |Q(s'', a'') - Q(s, a)| + |Q^\star(s, a) - Q(s, a)| \\ &\leq 2Kd'' + D. \end{aligned}$$

Using this as a bound on B and invoking Lemma A.1. demonstrates the subclaim.

Now, consider some arbitrary trajectory $s_0, a_0, s_1, a_2, \ldots, s_{k-1}, a_{k-1}, s_k$, with $s_0 \in \mathcal{S}_0$ and some fixed non-negative integer k. Via repeated application of the Triangle Inequality, one can bound the distance between s and s_k to at most $d_0 + kd_{\mathcal{T}}$. Invoking the subclaim, we see that

$$V^\star(s_k) - Q^\star(s_k, \pi^\star_{\text{backup}}(s_k)) \leq 2\gamma^n \left(D + 2K\sqrt{d_{\mathcal{A}}^2 + (d_0 + kd_{\mathcal{T}} + nd_{\mathcal{T}})^2} \right) + \epsilon_m.$$

We can repeatedly use the fact that $k + n \geq 1$ to clean up the bound here. Namely, we write

$$\begin{aligned} &D + 2K\sqrt{d_{\mathcal{A}}^2 + (d_0 + kd_{\mathcal{T}} + nd_{\mathcal{T}})^2} \\ \leq &D(k+n) + 2K\sqrt{(k+n)^2 d_{\mathcal{A}}^2 + ((k+n)d_0 + kd_{\mathcal{T}} + nd_{\mathcal{T}})^2} \\ &= (k+n)\left[D + 2K\sqrt{d_{\mathcal{A}}^2 + (d_0 + d_{\mathcal{T}})^2} \right]. \end{aligned}$$

Setting $C = D + 2K\sqrt{d_{\mathcal{A}}^2 + (d_0 + d_{\mathcal{T}})^2}$, we can sum up the equations above by writing

$$V^\star(s_k) - Q^\star(s_k, \pi^\star_{\text{backup}}(s_k)) \leq 2C\gamma^n(k+n) + \epsilon_m.$$

Now, we can finally bound RR. Let $\rho^{\pi_{\text{dmps}}}$ be the discounted state visitation distribution. We define $\rho_t^{\pi_{\text{dmps}}}$ to be the state visitation distribution at timestep t. We get

$$\begin{aligned} RR(\pi^\star_{\text{backup}}, \pi_{\text{dmps}}) &= \mathbb{E}_{s\sim\rho^{\pi_{\text{dmps}}}}[\mathbb{I}_{backup} \cdot (V^\star(s) - Q^\star(s, \pi^\star_{\text{backup}}(s)))] \\ &\leq \mathbb{E}_{s\sim\rho^{\pi_{\text{dmps}}}}[V^\star(s) - Q^\star(s, \pi^\star_{\text{backup}}(s))] \\ &= \int_{\mathcal{S}} [V^\star(s) - Q^\star(s, \pi^\star_{\text{backup}}(s))] \rho^{\pi_{\text{dmps}}}(s) ds \\ &= \int_{\mathcal{S}} [V^\star(s) - Q^\star(s, \pi^\star_{\text{backup}}(s))] \left[(1-\gamma) \sum_{k=0}^{\infty} \gamma^k \rho_k^{\pi_{\text{dmps}}}(s) \right] ds \\ &= \sum_{k=0}^{\infty} (1-\gamma)\gamma^k \int_{\mathcal{S}} [V^\star(s) - Q^\star(s, \pi^\star_{\text{backup}})] \rho_k^{\pi_{\text{dmps}}}(s) ds \\ &= \sum_{k=0}^{\infty} (1-\gamma)\gamma^k \mathbb{E}_{s\sim\rho_k^{\pi_{\text{dmps}}}}[V^\star(s) - Q^\star(s, \pi^\star_{\text{backup}}(s))] \end{aligned}$$

Algorithm 2 Monte Carlo Tree Search (`planRec`)

Inputs: (s_1: Start State), (Q: state-action function)
Hyper-parameters: (K: Branching Factor), (I: Iteration Count), (L: Maximum Path Length)
$\hat{Q} :=$ `initHashmap`$(\mathcal{S} \times \mathcal{A} \to \mathbb{R})$ # Initialize empty data structure for state-action value estimates.
$N :=$ `initHashmap`$(\mathcal{S} \times \mathcal{A} \to \mathbb{N})$ # Initialize empty data structure for state-action visit counts.
$T :=$ `initTree`(s_1) # Initialize a tree with start state s_0 as the root.
$res :=$ `expand`$(s_1, T, \hat{Q}, N)$ # Attempt expanding the tree root with actions leading to recoverable states.
if $\neg res$ **then return** $\perp$ # Return $\perp$ if the root state cannot be expanded.
for $t \in [1, I]$ **do**
  # Select a path of length less than L from the root using the Upper Confidence Bound (UCB) formula.
  $(s_1, a_1, R_1), \ldots, (s_{r-1}, a_{r-1}, R_{r-1}), s_r :=$ `selectPathUCB`$(\hat{Q}, L)$
  if `isLeaf`(s_r, T) **then**
    `expand`$(s_r, T, \hat{Q}, Q, N)$ # Expand the leaf state at the end of the path with actions leading to recoverable states.
  $a_r :=$ `selectAction`$(s_r, \hat{Q})$ # Choose the best outgoing action from s_r with respect to $\hat{Q}$.
  for $i \in [2, r)$ **do**
    # Update the state-action estimates with rewards collected on the selected path.
    $\hat{Q}(s_i, a_i) := \frac{1}{N(s_i,a_i)+1} \left[N(s_i, a_i) \cdot \hat{Q}(s_i, a_i) + \left(\left(\sum_{j=i}^{r-1} \gamma^{j-i} \cdot R_j \right) + \gamma^{r-i} \cdot \hat{Q}(s_r, a_r) \right) \right]$
  for $i \in [2, r)$ **do**
    $N(s_i, a_i) := N(s_i, a_i) + 1$ # Increment state-action visit count for the selected path.
$(s_1, a_1), (s_2, a_2), \ldots, (s_n, a_n) :=$ `selectPathGreedy`$(\hat{Q})$ # Use $\hat{Q}$ to greedily select the optimal path from root to a leaf.
return $(a_1, \ldots, a_n)$ # Return the actions on the greedily selected path.

Algorithm 3 Sampling-Based Tree Expansion (`expand`)

Inputs: (s: Expanding State), (T: State-Action Tree), ($\hat{Q}$: Local Value Estimate), (Q: Long-term Value Function), (N: Visit Count)
$a_1, \ldots, a_K \sim \mathcal{A}$ # Sample K actions.
$success := False$
for $i \in [1, K]$ **do**
  if $\mathcal{T}(s, a_i) \in \mathcal{S}_{rec}$ **then**
    $\hat{Q}(s, a_i) := Q(s, a_i)$ # Initialize value estimate for (s, a_i).
    $N(s, a_i) := 1$ # Record first visit for (s, a_i) pair.
    `updateTree`(T, s, a_i) # Add (s, a_i) to the tree.
    $success := True$ # Record at least one recoverable action was found.
return $success$

$$\leq \sum_{k=0}^{\infty} (1-\gamma)\gamma^k [2C\gamma^n(k+n) + \epsilon_m]$$

$$= \left[2C(1-\gamma)\gamma^n \sum_{k=0}^{\infty} \gamma^k (k+n) \right] + \sum_{k=0}^{\infty} (1-\gamma)\gamma^k \epsilon_m$$

$$= 2C(1-\gamma)\gamma^n \left[\left(\sum_{k=0}^{\infty} k\gamma^k \right) + n \sum_{k=0}^{\infty} \gamma^k \right] + \epsilon_m.$$

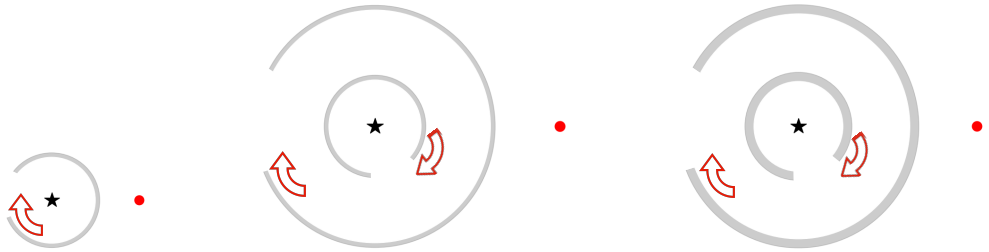
We can evaluate the arithmetic-geometric series $\sum_{k=0}^{\infty} k\gamma^k$ to equal $\frac{\gamma}{(1-\gamma)^2}$. The geometric sequence $\sum_{k=0}^{\infty} \gamma^k$ evaluates to $\frac{1}{1-\gamma}$. Thus, our sum becomes

$$2C(1-\gamma)\gamma^n \left[\frac{\gamma}{(1-\gamma)^2} + \frac{n}{1-\gamma} \right] + \epsilon_m = \mathcal{O}(n\gamma^n) + \epsilon_m.$$

Due to asymptotic optimality, taking the limit of this as m goes to infinity gives a bound of $\mathcal{O}(n\gamma^n)$. □

A.2 Monte Carlo Tree Search

The planning function `planRec` used in `DMPS` is realized via Monte Carlo Tree Search (MCTS) [79], which facilitates efficient online planning by balancing exploration and exploitation. MCTS has been shown to be highly effective for planning in large search spaces with either discrete [52, 81, 51] or continuous [54, 57, 82] states and actions. `DMPS` utilizes a continuous variant of MCTS that employs sampling-based methods to manage the exponential increase in search space size as the planning horizon expands [53]. A high-level overview of this approach follows. Algorithms 2 and 3 present our implementation of MCTS.



(a) `gate` (b) `double-gates` (c) `double-gates+`

Figure 6: Visualization of the dynamic environments. The agent is depicted as a red circle. The direction of rotation of the walls is shown with red arrows. The goal position is shown with $\star$.

Starting from a given state s_0 as the root node, the function `planRec` maintains and iteratively expands a tree structure, with states as nodes and actions as edges. During each iteration, the algorithm selects a path from the root to a leaf node. Upon reaching a leaf node, it extends the tree by sampling a number (k) of actions from the leaf, thereby adding new nodes and edges to the tree. The path selection process is based on the Upper Confidence Bound (UCB) formula [83], which utilizes an internal representation of the state-action values at each tree node (denoted by $\hat{Q} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$), to balance exploration of less frequently visited paths with exploitation of paths known to yield high returns. Once a path is selected, the algorithm updates the value of $\hat{Q}$ for each node on the selected path by backpropagating the accumulated rewards from subsequent nodes. These updates are averaged by each state-action's visit count to achieve more precise estimates.

After a specified number of iterations (I), the algorithm uses $\hat{Q}$ to greedily select an optimal path from the root to a leaf node and returns the corresponding sequence of actions as its final result.

A.3 Additional Experimental Results

A.3.1 Benchmarks

A detailed description of benchmarks used in our evaluation is presented below:

- `obstacle`: This benchmark involves a robot moving on a 2D plane, aiming to reach a goal position without colliding with a stationary obstacle. This obstacle is positioned to the side, impacting the robot only during its exploration phase and not on the direct, shortest route to the goal.
- `obstacle2`: This benchmark involves a robot moving on a 2D plane, aiming to reach a goal position without colliding with a stationary obstacle. This obstacle is positioned between the starting point and the goal, requiring the robot to learn how to maneuver around it.
- `mount-car`: This benchmark requires moving an underpowered vehicle up a hill without falling off the other side.
- `road`: This benchmark requires controlling an autonomous car to move in one dimension to a specified end position while obeying a given speed limit.
- `road2d`: This benchmark requires controlling an autonomous car to move in two dimensions to a specified end position while obeying a given speed limit.
- `dynamic-obs`: This benchmark features multiple non-stationary obstacles that move deterministically in small circles on the path from the agent to the goal.
- `single-gate`: In this benchmark, the goal position is surrounded by a circular wall with a small opening, allowing the agent to pass through. The position of the opening continuously rotates around the goal position. This is visualized in Fig. 6a.
- `double-gates`: This benchmark is similar to `single-gate`; however, the goal position is surrounded by two concentric rotating walls. This is visualized in Fig. 6b.
- `double-gates+`: This benchmark is similar to `double-gates`; however, the thickness of the rotating walls is increased. This poses a greater challenge, as the agent has a shorter

time window to cross through the opening without colliding with the rotating wall. This is visualized in Fig. 6c.

For the dynamic benchmarks above (`dynamic-obs`, `single-gate`, `double-gate`, and `double-gate+`), the action space of the double integrator agent consists of acceleration in the x and y directions. The action space for the differential drive agent consists of the torque value on the left wheel and the torque value on the right wheel.

Additionally, the observation space of benchmarks with a double integrator agent consists of the position and velocity vectors of the agent, while for a differential drive agent, the observation space includes position, velocity, and pose angle. The observation space for `single-gate`, `double-gate`, and `double-gate+` also contains an angle term for each wall, corresponding to the current rotation of each wall. The observation space in `dynamic-obs` additionally includes an angle term for each obstacle, indicating how far each obstacle has moved along its circular trajectory.

A.3.2 Implementation

We implemented `DMPS` by modifying the Twin Delayed Deep Deterministic Policy Gradient (TD3) algorithm [15], following the description presented in Algorithm 1.

Each of the dynamic benchmarks were trained for 200,000 timesteps with a maximum episode length of 500. The static environments were trained for the number of timesteps prescribed by the sources of the environments. Namely, `mount-car` was trained for 200,000 timesteps with a maximum episode length of 999 [84], `obstacle` and `obstacle2` were trained for 400,000 timesteps with a maximum episode length of 200 [16], and `road` and `road2d` were trained for 100,000 timesteps with a maximum episode length of 100 [16]. Each experiment was run on five independent seeds. We used prior implementations of `REVEL` [16], `PPO-Lag` [85], and `CPO` [85] to run our experiments. Trained models were test evaluated every 10K timesteps, independently from the training loop. Each test evaluation consisted of 10 runs. The final test evaluation was used to determine the average per-episode return and per-episode shield invocation rate for that random seed. These are the values displayed in the tables from section 6.

The reward functions used in [16] for the static environments are non-standard, and based upon a cost-based framework of reward. In light of this, we used the canonical reward functions for our evaluations. For `mount-car`, we use the original reward function defined in [84]. For `obstacle`, `obstacle2`, `road`, and `road2d`, we use the canonical goal environment shaped reward function.

The negative penalty for safety violations in `TD3` was taken to be large enough so that the agent could not move through obstacles and still maintain positive reward. In most cases, the penalty was simply the negation of the positive reward incurred upon successfully completing the environment. The episode did not terminate upon the first unsafe action. For `CPO`, we reduced tolerance for safety violations by reducing the parameter for number of acceptable violations to 1.

Our experiments were conducted on a server with 64 available Intel Xeon Gold 5218 CPUs @2.30GHz, 264GB of available memory, and eight NVIDIA GeForce RTX 2080 Ti GPUs. Each benchmark ran on one CPU thread and on one GPU.

A.4 Analysis of Computational Overhead

As mentioned in the main text, our implementation uses MCTS in an anytime planning setting, where we devote a fixed number of node expansions to searching for a plan, and we choose the best plan found at the end of the procedure. This bounds the clock time needed for planning linearly in the compute budget allocated. However, in the worst case, one may need a compute budget exponential in size with respect to the planning horizon H if one wishes to explore the planning state densely. This is common to all general planners.

Here, we try to experimentally evaluate how the required compute budget scales as a function of the planning horizon. We re-evaluate the `double-gates+` environment (under double integrator dynamics). For each planning horizon H in range $[2, 9]$, we count the average number of node expansions that MCTS needs before it successfully explores 10 states at depth H. We use this as a proxy for successful exploration of the horizon H search space. The results of this experiment can be found in Figure 7. As expected, an exponential relationship emerges.

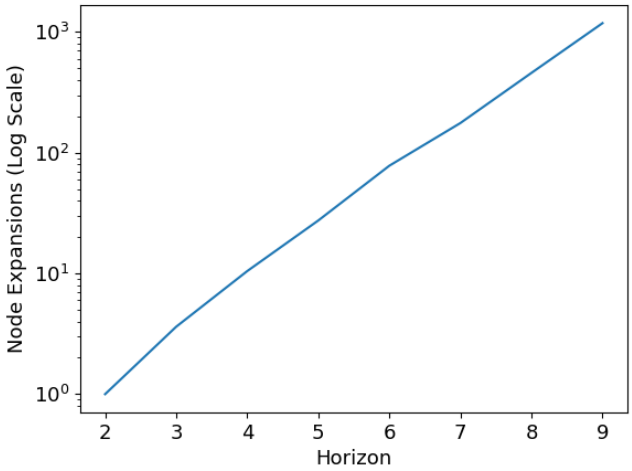


Figure 7: Planner Computation Scaling

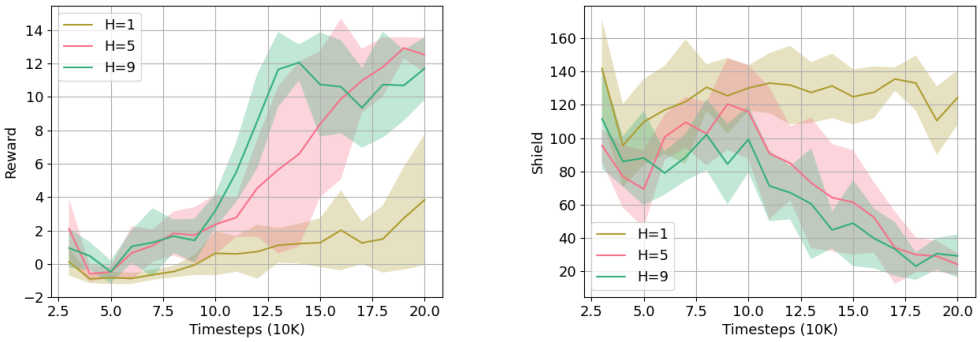


Figure 8: Multiple Horizons Experiments

A.5 Effect of Planning Horizon

We expand on the question of whether small planning horizons are sufficient to solve tricky planning problems. Our algorithm is designed to ensure that the planner accounts for both short-term and long-term objectives. As detailed in section 5, the objective of the planner consists of two terms: 1) the discounted sum of the first n rewards, and 2) the estimated long-term value of the penultimate step in the plan horizon, as determined by the agent's Q function. The second part of the objective function is specifically included to ensure that the planner does not return myopic plans, and accounts for progress towards the long-horizon goal. Since the planner optimization objective includes this second term, even a small-horizon planner can output actions with proper awareness of long-horizon recovery events. The length of the horizon affects how close to the globally optimal solution the result is, with a tradeoff of computational cost, as was established in Theorem 5.1.

To see this empirically, we reevaluated the `double-gates+` environment (double integrator dynamics) with horizons of 1, 5, and 9. The graphs of attained reward and shield triggers from this experiment are shown in Figure 8.

Comparing the $H = 1$ and $H = 5$ agents, the $H = 5$ agent substantially outperforms the $H = 1$ agent in both shield triggers and reward. Comparing $H = 5$ and $H = 9$, the $H = 9$ agent reached high performance and low shield triggers faster than the $H = 5$ agent. However, both the $H = 5$ and $H = 9$ agents converge to the same performance eventually.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Our main claim in this paper is that MPS, a state-of-the-art framework for Provably Safe RL, has a significant shortcoming due to its inability to recover from potentially unsafe situations while making task progress. We introduce a solution for this problem and claim that it adequately resolves the shortcoming of MPS. Throughout the paper, we extensively elaborate on the nature of MPS's shortcoming. We present our solution based on a local planner and provide a theoretical analysis explaining the asymptotic optimality of our solution. Our experiments validate our claims that our approach achieves superior performance compared to MPS. We also show that our approach is superior to several other well-known SRL and PSRL methods.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We present a discussion of the limitations of our approach in section 8. In summary, our approach requires an accurate world model, and the computational overhead of planning increases exponentially with the planning horizon.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.

- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: In this paper, we present make theoretical contribution, which is presented in Theorem 5.1. We have included a complete, correct and detailed proof for this theorem in Appendix A.1.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We formally define all objectives solved by our solution and present detailed algorithms implementing the main approach in Algorithms 1, 2, and 3. Our submission also includes the source code of our implementations and scripts for reproducing the results. We also plan to create an open-source repository and make our implementation publicly available.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example

(a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
(b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We provide the code necessary to run our own method (DMPS) and MPS, as well as implementations of the environments on which the algorithms were run. All other baselines are publically available.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: In Appendix A.3.2 we present the specifics of our implementation and experimental setup in detail. This includes the hyperparameters, the training and test set details, the number of runs, and the hardware used for experimentation.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.

- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The statistical significance of all presented empirical results is justified. In particular, we include standard deviation values for all results presented in Table 1 and Table 2. Additionally, we include confidence belts in all plots presented in Figure 3 and Figure 4.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We present the details of the hardware and other resources used for our experiments in Appendix A.3.2.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

Justification: We have reviewed the NeurIPS Code of Ethics carefully and affirm that our work adheres to all listed requirements.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: To the best of our knowledge, our work cannot be misused in any way to cause any form of negative social impact. We have included a discussion of the potentially positive social impacts of our work in section 8.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: To the best of our knowledge, our work does not pose a risk for misuse.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.

- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We have only used publicly available open-source libraries in our implementation and credited all third-party sources accordingly.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: We do not release any new assets with this paper.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.

- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: This paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.