SAFER: Data-Efficient and Safe Reinforcement Learning via Skill Acquisition

Anonymous Authors[1]

Abstract

Though many reinforcement learning (RL) problems involve learning policies in settings with difficult-to-specify safety constraints and sparse rewards, current methods struggle to acquire successful and safe policies. Methods that extract useful policy primitives from offline datasets using generative modeling have recently shown promise at accelerating RL in these more complex settings. However, we discover that current primitive-learning methods may not be well-equipped for safe policy learning and may promote *unsafe* behavior due to their tendency to ignore data from undesirable behaviors. To improve the safety of offline skill learning algorithms, we propose *SAFEty skill pRiors (SAFER)*, an algorithm that accelerates policy learning on complex control tasks under safety constraints. Through principled training on an offline dataset, SAFER learns to extract *safe primitive skills*. In the inference stage, policies trained with SAFER learn to compose safe skills into successful policies. We theoretically characterize why SAFER can enforce safe policy learning and demonstrate its effectiveness on several complex safety-critical robotic grasping tasks inspired by the game Operation,[1] in which SAFER outperforms baseline methods in learning successful policies and enforcing safety.

1. Introduction

Reinforcement learning (RL) has demonstrated impressive performance at solving complex control tasks. However, RL algorithms still require considerable exploration to acquire successful policies. For many complex safety-critical applications (i.e., autonomous driving, healthcare, factory robotics), extensive interaction with an environment is impossible due to potential dangers associated with exploration. These difficulties are further complicated because it is challenging to specify safety constraints manually because of the complexity of environments. Nevertheless, relatively few existing safe reinforcement learning algorithms can *rapidly*, and *safely* solve high-dimensional RL problems with challenging to specify safety constraints.

One promising route is offline primitive learning methods (Singh et al., 2021; Pertsch et al., 2020; 2021; Ajay et al., 2021). These methods use offline datasets to learn representations of useful actions or *behaviors* through generative models, such as normalizing flow models or variational autoencoders (VAE). Specifically, they treat the latent space of the generative model as the abstract action space of higher-level actions (i.e., skills). These methods train an RL agent to map states onto the abstract action space of skills for each downstream task using the learned primitives. This approach can significantly accelerate policy learning because it first learns useful primitives from a dataset, simplifying the action space (Dulac-Arnold et al., 2015).

Intuitively, if trained on datasets consisting of trajectories that are both safe and successful, offline skill learning methods should capture *safe and useful* behaviors and encourage the *rapid* acquisition of safe policies on future tasks (*downstream* learning). For example, when trained on data from everyday household tasks, these methods should learn behaviors that successfully and safely accomplish similar tasks, such as handling objects carefully or avoiding animals in the environment. However, offline skill learning methods may not be well suited for safe policy learning. This result is because current state-of-the-art methods cannot distinguish between safe and unsafe actions and are unable to avoid the concentration of unsafe actions in high likelihood regions of the abstract action space (see Figure 1). When offline skill learning methods (oftentimes modeled with deep generative models) are trained only with safe experiences, the unsafe data is out of the training distribution. It is well known that deep generative models have problems generalizing to out of distribution data, which indeed increases the likelihood of unsafe actions in this case (Nalisnick et al., 2018; Fetaya et al., 2020; Kirichenko et al., 2020). For the household robotics example, the learned primitives may lead to unsafe behavior (breaking objects or harming animals/people).

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

[1]https://en.wikipedia.org/wiki/OperationGame

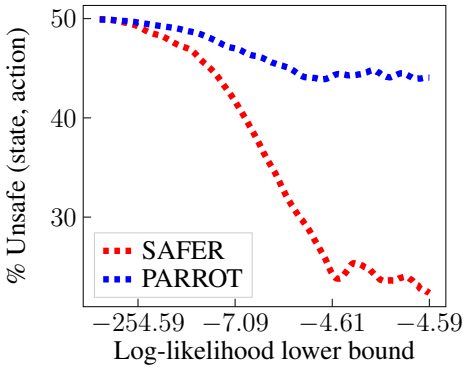

Figure 1. **Evaluating the concentration of unsafe data in high likelihood regions** by computing the % of unsafe state-action pairs in a holdout dataset of a safe robotic grasping task. PARROT, a state of the art offline skill learning method (Singh et al., 2021), assigns high likelihoods to unsafe data, i.e., it does not encourage safety, while SAFER has much lower likelihood in unsafe data.

In this work, we propose *SAFER*: safety skill priors, which possesses both desiderata: *accelerating* reinforcement learning with *safe* operations. (An overview of SAFER is provided in Figure 2.) To acquire safe offline skill-priors, SAFER **i)** uses a *contrasive loss* to distinguish safe date from unsafe ones and **ii)** learns a posterior sampling distribution of a latent *safety variable*, that captures different *safety contexts*. With this in hand, SAFER maps the abstract space onto safe policy primitives. These safe actions are parameterized by the safety context variable, which makes the technique more adaptable to different tasks and safety constraints. Consequently, policies trained using the SAFER abstract actions as the action space will learn to compose a set of safe policy primitives. To further establish the safety assurance of SAFER, we propose a technique to adjust the abstract action space, such that at any state s, at most $(1-\epsilon)\%$ of actions generated by the safety prior is unsafe. As shown in Figure 1, SAFER assigns much lower likelihood to unsafe states and actions, indicating that it will better promote safe behaviors when applied to downstream RL. To demonstrate the effectiveness of SAFER, we evaluate it on a set of complex safety-critical robotic grasping tasks. When compared with other baseline methods, SAFER has a higher success rate and fewer safety violations.

2. Background

In a setting with different tasks, for each task $\mathcal{T}$, the agent's interaction is modeled as a Markov decision process (MDP). A MDP is a tuple $(\mathcal{S}, \mathcal{A}, \mathrm{T}, r, \gamma, \boldsymbol{s}_0)$. In the MDP, $\mathcal{S}$ and $\mathcal{A}$ are the state and action spaces, $\mathrm{T}(\cdot|\boldsymbol{s}, \boldsymbol{a})$ is the transition probabilities, $r(\boldsymbol{s}, \boldsymbol{a})$ is the reward function, $\gamma \in [0, 1)$ is the discount factor, and $\boldsymbol{s}_0 \in \mathcal{S}$ is the initial state. A number of different safe RL formulations exist in the literature (García et al., 2015). In this work, safety is measured by the *safety violation function*, $\omega(\boldsymbol{s}, \boldsymbol{a}) \in \{0, 1\}$, that indicates whether the current state and action lead to a safety violation (1) or no safety violation (0). Consequently, the safety MDP is given as a tuple $(\mathcal{S}, \mathcal{A}, \mathrm{T}, r, \gamma, \boldsymbol{s}_0, \omega)$. Note, that the safety violation function $\omega(\boldsymbol{s}, \boldsymbol{a})$ is task specific. This formulation is desirable because different tasks likely have different restrictions, and thus, different safety criterion. For instance, in robotic grasping, different tasks likely contain different objects that have various sensitivities to being gripped or limits to how quickly they can move without breaking. Notice that we utilize a binary notion of safety violation because we are concerned with complex RL problems, (e.g., learning from pixels), where it can be difficult to specify continuous notions of safety. We assume we don't have access to the safety violation function during training, and only have access to state action pairs labeled as safe or unsafe. We only use the safety violation function during evaluation.

We now formalize the safety MDP problem for each task in the environment. Let Δ be the set of Markovian stationary policies, i.e., $\Delta = \{\mu : \mathcal{S} \times \mathcal{A} \to [0, 1],\ \sum_{\boldsymbol{a}} \mu(\boldsymbol{a}|\boldsymbol{s}) = 1\}$. Given a policy $\mu \in \Delta$, we define the expected return as $\mathcal{R}_\mu(\boldsymbol{s}_0) := \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r(\boldsymbol{s}_t, \boldsymbol{a}_t) \mid \mu, \boldsymbol{s}_0]$ and at each given state $\boldsymbol{s} \in \mathcal{S}$ the safety constraint function (i.e., expected safety violation) as $\mathcal{W}_\mu(s) := \mathbb{E}[\omega(\boldsymbol{s}, \boldsymbol{a}) \mid \mu, \boldsymbol{s}]$. The *safety constraint* is then defined as $\mathcal{W}_\mu(s) \leq \epsilon$, where $\epsilon \in [0, 1]$ is the tolerable threshold of violation. For each task the goal in safety MDP is to solve the constrained optimization problem

$$\mu^* \in \arg\max_{\mu \in \Delta} \{\mathcal{R}_\mu(\boldsymbol{s}_0) : \mathcal{W}_\mu(\boldsymbol{s}) \leq \epsilon,\ \forall \boldsymbol{s} \in \mathcal{S}\}. \quad (1)$$

It has been shown that if the feasibility set is non-empty, then there exists an optimal policy in the class of stationary Markovian policies Δ (Altman, 1999, Theorem 3.1). Similar to policy gradient (PG) algorithms, to effectively solve for the optimal policy we parameterize the stationary Markovian policy by a κ-dimensional vector ψ, so the space of policies can be written as $\{\mu_\psi,\ \psi \in \Psi \subset \mathbb{R}^\kappa\}$. In Section 3.2, we exploit the connections of different tasks to design a parameterization structure with policy primitives that leads to effective and safe learning.

Offline Primitive Learning To effectively solve the safety MDP for each task, similar to the recent work of offline *primitive discovery* (Singh et al., 2021; Ajay et al., 2021; Pertsch et al., 2020; 2021), we propose the specific policy structure $\mu_\psi = f_\phi(\boldsymbol{z}; \boldsymbol{s})$ and $\boldsymbol{z} \sim \pi_\theta(\boldsymbol{z}|\boldsymbol{s})$, where $\psi = (\phi, \theta)$. In this parameterization, the mapping $f_\phi : \mathcal{Z} \times \mathcal{S} \to \mathcal{A}$ with learnable parameters ϕ, sometimes called the *behavioral prior*, maps from the abstract action space $\mathcal{Z}$ and state space $\mathcal{S}$ to the action space $\mathcal{A}$. The task-dependent, high-level policy $\pi_\theta : \mathcal{S} \to \mathbb{P}(\mathcal{Z})$ maps any state $\boldsymbol{s} \in \mathcal{S}$ to the corresponding distribution of abstract actions in $\mathcal{Z}$. Notice that the prior f_ϕ is independent of any tasks. To train this action mapping more effectively, one

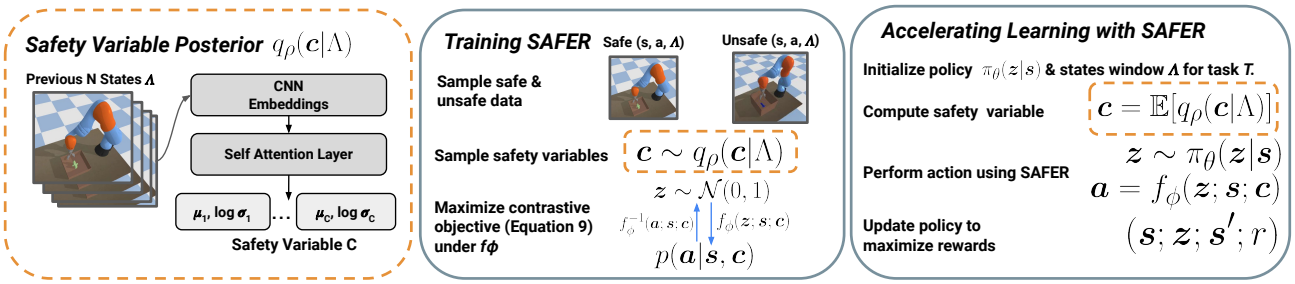

Figure 2. **Overview of SAFER:** SAFER optimizes the posterior over a latent *safety variable* (left hand side of figure) that encodes safety information of the environment. SAFER uses the safety variable to learn an abstract action space $\mathcal{Z}$ that maps to safe and useful behaviors through f_ϕ through a normalizing flow (middle of figure). SAFER accelerates RL training by learning a latent-action policy $\pi_\theta(\boldsymbol{z}|\boldsymbol{s})$ in $\mathcal{Z}$ (right hand side of figure).

can use an offline dataset $\mathcal{D}$, which consists of state-action rollouts $\tau = \{\boldsymbol{s}_0, \boldsymbol{a}_0, ..., \boldsymbol{s}_t, \boldsymbol{a}_t\}$ collected across different tasks. As long as these rollouts are generated by following a diverse set of policies, they will likely contain information for learning different low-level action mappings from $\mathcal{Z} \times \mathcal{S}$ to $\mathcal{A}$ that leads to useful behavior for the downstream tasks. Typically, a simple distribution is chosen as $\mathcal{Z}$ (e.g., the unit normal distribution) to make controlling the primitives straightforward.

Policies trained in this way use $\mathcal{Z}$ as the action space and optimize the parameters θ in the policy π_θ. In the context of the safety MDP, the learned primitives provide a simplified action space $\mathcal{Z}$ which only contains actions in the action space $\mathcal{A}$ that will satisfy the safety threshold $\mathcal{W}_\mu(s) := \mathbb{E}[\omega(\boldsymbol{s}, \boldsymbol{a}) \mid \mu, \boldsymbol{s}]$ and maximize rewards $\mathcal{R}_\mu(\boldsymbol{s}_0)$. For example, in a robotic grasping task, the mapping f_ϕ may learn to reach toward objects or grip nearby objects. These properties make offline primitive learning techniques highly appealing for solving complex control tasks because learning successful policies from high level behaviors typically requires much less interaction with the environment than learning policies from scratch (Singh et al., 2021). As such, different ways to express the behavior prior mapping have been proposed. For instance, Ajay et al. (2021) optimizes the likelihood of actions, conditioned on the state and abstract action space, $\log \pi_\theta(\boldsymbol{a}|\boldsymbol{s}, \boldsymbol{z})$. Singh et al. (2021) directly optimizes the log-likelihood policy, $\log p(\boldsymbol{a}|\boldsymbol{s})$, and fix an invertible mapping through the use of a conditional normalizing flow (Dinh et al., 2017) between the abstract action space $\mathcal{Z}$ and the distribution over useful actions $p(\boldsymbol{a}|\boldsymbol{s})$ to retain control of the learned behaviors.

Issues With Offline Primitive Discovery for Safe RL Though current offline primitive discovery methods are highly useful at accelerating learning, they only *increase* the likelihood of useful actions. Thus, when applied to a safety MDP problem, data containing unsafe or unsuccessful data should not be used because it is counter-intuitive to increase the likelihood of these actions (Singh et al., 2021; Ajay et al., 2021). Consequently, unsafe states and actions may be out of distribution (OOD). It is well established in the literature on deep generative models (including the techniques used in offline primitive discovery methods) that OOD data is handled poorly and, in some cases, might have higher likelihood than in-distribution data (Nalisnick et al., 2018; Fetaya et al., 2020; Kirichenko et al., 2020). As we see in Figure 1, these observations hold true for current techniques where unsafe data has high likelihood, indicating that they may encourage unsafe behavior in the presence of OOD states. Since the proposed offline primitive discovery policy structure relies on high likelihood actions from the prior (Ajay et al., 2021; Singh et al., 2021), using the aforementioned behavior priors to solve safety MDP problems will generally be problematic.

3. SAFER: Safety Skill Priors

Considering the shortcomings mentioned in Section 2 of existing offline primitive discovery techniques and the need for methods that can learn complex safety constraints, ideally a method that encourages safety should **i)** be capable of learning complex safety constraints by sufficiently exploiting the data, thereby avoiding the OOD issue; **ii)** permit the specification of undesirable behaviors through data; and **iii)** accelerate the learning of successful policies. Motivated by these requirements, in this section we introduce SAFER, an offline primitive learning method that circumvents the aforementioned shortcomings and is specifically designed for safety MDPs.

3.1. Latent Safety Variable

To address all of the above criteria, we introduce an additional latent variable called the *safety variable* $\boldsymbol{c} \in \mathcal{C}$ that encodes safety information about the current environment, i.e., $f_\phi : \mathcal{Z} \times \mathcal{C} \times \mathcal{S} \to \mathcal{A}$. This construction is useful because information beyond the current state $\boldsymbol{s}$ can be encoded into the task variable, to better help SAFER model complex per task safety dynamics. For example, the safety variable

could encode the locations of other vehicles in autonomous driving or animals/people in a robotics application. Because we do not assume the task variable $\mathcal{C}$ is provided from the environment, we infer it from a network.

3.2. Learning The Safety Variable

In order to train the prior f_ϕ and posterior over the safety variable, we adopt a variational inference (VI) approach. We jointly train an invertible conditional normalizing flow f_ϕ (Dinh et al., 2017) as the prior f_ϕ and posterior over the safety variable using VI. At each state $s \in \mathcal{S}$ and safety variable $c \in \mathcal{C}$, the flow model f_ϕ maps a unit Normal abstract action $z \in \mathcal{Z}$ (i.e., samples $z = f_\phi^{-1}(\boldsymbol{a}|\boldsymbol{s}, \boldsymbol{c})$ of the inverse flow model follow the distribution $p_{\mathcal{Z}}(\cdot) := \mathcal{N}(0, I)$) onto the action space $\mathcal{A}$ of safe behaviors, and thus, the corresponding prior action distribution is given by

$$p_\phi(\boldsymbol{a}|\boldsymbol{s}, \boldsymbol{c}) := p_{\mathcal{Z}}(f_\phi^{-1}(\boldsymbol{a}; \boldsymbol{s}; \boldsymbol{c})) \cdot |\det(\partial f_\phi^{-1}(\boldsymbol{a}; \boldsymbol{s}; \boldsymbol{c})/\partial \boldsymbol{a})|. \tag{2}$$

The flow model is a good choice for the mapping f_ϕ because it allows computing exact log likelihoods. Further, it yields a mapping such that actions taken in the abstract action space $\boldsymbol{z} \in \mathcal{Z}$ can easily be transformed into useful ones $\boldsymbol{a} = f_\phi(\boldsymbol{z}; \boldsymbol{s}; \boldsymbol{c})$. However, since VI approximates the lower bound of maximum likelihood, it does not explicitly enforce the safety requirements in the safety variable $\boldsymbol{c}$. To overcome this issue, we encode safety to $\boldsymbol{c}$ by formulating the learning problem as a chance constrained optimization (Charnes & Cooper, 1959) problem.

Chance Constrained Optimization Formally, our objective arises from optimizing a neural network to infer the posterior over the safety variable $\mathcal{C}$ using amortized variational inference (Zhang et al., 2019). In particular, we parameterize the posterior over the safety variable as $q_\rho(\boldsymbol{c} \,|\, \Lambda)$, where $\boldsymbol{c}$ is the safety variable, and Λ is information from which to infer the variable. In practice, we set Λ as a sliding window of states, such that if $\boldsymbol{s}_t$ is the current state at time t and w is the window size, then the information is given by $\Lambda = [\boldsymbol{s}_t, \boldsymbol{s}_{t-1}, ..., \boldsymbol{s}_{t-w}]$. We infer the safety variable from the sliding window of states Λ because we expect Λ to contain useful information concerning safe learning. For example, in a robotics setting where the observations are images, previous states may contain useful information concerning the locations of objects to avoid, which may be unobserved in the current state. Specifically, we write the evidence-lower bound (ELBO) of our model as

$$\mathbb{E}_{\boldsymbol{c} \sim q_\rho(\cdot|\Lambda)} \left[\log p_\phi(\boldsymbol{a}|\boldsymbol{s}, \boldsymbol{c})\right] - D_{\mathrm{KL}}(q_\rho(\cdot|\Lambda)||p(\cdot)), \tag{3}$$

where $\boldsymbol{a}$ is the *safe* action (i.e, $\omega(\boldsymbol{a}, \boldsymbol{s}) = 0$) and $p(\boldsymbol{c})$ is a prior over the safety variable $\boldsymbol{c}$. To ensure that SAFER only samples unsafe actions with low probability, we add a chance constraint about the likelihood of unsafe actions (Nemirovski & Shapiro, 2007) to the ELBO optimization,

$$\begin{aligned} \max_{\rho,\phi,\xi} \; & \mathbb{E}_{\boldsymbol{c} \sim q_\rho(\cdot|\boldsymbol{s})} \left[\log p_\phi(\boldsymbol{a}|\boldsymbol{s}, \boldsymbol{c})\right] - D_{\mathrm{KL}}\left(q_\rho(\cdot|\Lambda)||p(\cdot)\right) - \lambda' \xi \\ \text{s.t. } & \mathbb{P}_{\boldsymbol{c} \sim q_\rho(\cdot|\boldsymbol{s})}(p_\phi(\boldsymbol{a}_{\text{unsafe}}|\boldsymbol{s}, \boldsymbol{c}) > \epsilon) \leq \xi, \end{aligned} \tag{4}$$

where the constraint states that in most cases (at least with probability ξ) with the safety variable c drawn from $\mathcal{C}$ the distribution of the corresponding unsafe actions (i.e., $\omega(\boldsymbol{a}_{\text{unsafe}}, \boldsymbol{s}) = 1$) is always less than the safety threshold ϵ. Intuitively, this objective enforces that the safety variable should make safe actions as likely as possible while minimizing the probability of unsafe actions.

Tractable Lower Bound Due to the difficulty in optimizing the chance constrained ELBO objective, we instead consider optimizing an unconstrained surrogate lower bound (Nemirovski & Shapiro, 2007).

Proposition 3.1. *Assuming the chance constrained ELBO is written as in Equation 4, we can write the surrogate lower bound as,*

$$\begin{aligned} \max_{\rho,\phi} \; & \mathbb{E}_{\boldsymbol{c} \sim q_\rho(\cdot|\boldsymbol{s})} \Big[\log p_\phi(\boldsymbol{a}|\boldsymbol{s}, \boldsymbol{c}) - \lambda \log p_\phi(\boldsymbol{a}_{unsafe}|\boldsymbol{s}, \boldsymbol{c})\Big] \\ & - D_{KL}(q_\rho(\cdot|\Lambda)||p(\cdot)) \end{aligned} \tag{5}$$

Proof: We rewrite the optimization 4 into the following form,

$$\begin{aligned} \max_{\rho,\phi,\lambda'} \; & \mathbb{E}_{\boldsymbol{c} \sim q_\rho(\cdot|\boldsymbol{s})} \left[\log p_\phi(\boldsymbol{a}|\boldsymbol{s}, \boldsymbol{c}) - D_{\mathrm{KL}}(q_\rho(\cdot|\Lambda)||p(\cdot))\right] \\ & - \lambda' \mathbb{P}_{\boldsymbol{c} \sim q_\rho(\cdot|\boldsymbol{s})}(p_\phi(\boldsymbol{a}_{\text{unsafe}}|\boldsymbol{s}, \boldsymbol{c}) > \epsilon). \end{aligned} \tag{6}$$

With the Markov inequality we have

$$\mathbb{P}_{\boldsymbol{c} \sim q_\rho(\cdot|\boldsymbol{s})}(p_\phi(\boldsymbol{a}_{\text{unsafe}}|\boldsymbol{s}, \boldsymbol{c}) > \epsilon) \leq \frac{\mathbb{E}_{\boldsymbol{c}}[p_\phi(\boldsymbol{a}_{\text{unsafe}}|\boldsymbol{s}, \boldsymbol{c})]}{\epsilon}, \tag{7}$$

such that the following objective function is a lower bound of that in Equation 4:

$$\begin{aligned} \max_{\rho,\phi,\lambda'} \; & \mathbb{E}_{\boldsymbol{c} \sim q_\rho(\cdot|\boldsymbol{s})} \Big[\log p_\phi(\boldsymbol{a}|\boldsymbol{s}, \boldsymbol{c}) - \frac{\lambda'}{\epsilon} p_\phi(\boldsymbol{a}_{\text{unsafe}}|\boldsymbol{s}, \boldsymbol{c})\Big] \\ & - D_{\mathrm{KL}}(q_\rho(\cdot|\Lambda)||p(\cdot)) \end{aligned} \tag{8}$$

For convenience, we write $\frac{\lambda'}{\epsilon}$ as the single hyperparameter λ and optimize the log of $p_\phi(\boldsymbol{a}_{\text{unsafe}}|\boldsymbol{s}, \boldsymbol{c})$ for better numerical stability. We finally have the lower bound surrogate objective in Equation 5.

We denote this objective as the *SAFER Contrastive Objective*. Beyond rigorous derivations, this ELBO objective function is intuitively interpretable. The first two terms together act as a contrastive loss that *encourages* safe actions (high likelihood) while *discourages* unsafe ones (low likelihood). Together with the final term, the task variable $\boldsymbol{c}$ is then forced to only contain useful information that does not violate safety. Thus the objective satisfies the earlier goals,

Algorithm 1 Accelerating Safe Reinforcement Learning with SAFER

Require: SAFER Prior f_ϕ, Safety Posterior $q_\rho(\boldsymbol{c}|\Lambda)$, Safety bound η, Task $\mathcal{T}$, Window $\Lambda = \{\}$
  for step $k = 1, ..., K$ **do**
    $\boldsymbol{s}_k \leftarrow$ current state
    $\boldsymbol{c}_k \leftarrow \mathbb{E}_{\boldsymbol{c}\sim q_\rho(\cdot|\Lambda_k)}[\boldsymbol{c}]$ {`Mean safety var.`}
    $\boldsymbol{z}_k \sim \pi_\theta(\cdot|\boldsymbol{s}_k)$ {`Sample abstract action`}
    $\boldsymbol{a}_k \leftarrow f_\phi(\boldsymbol{z}_k; \boldsymbol{s}_k; \boldsymbol{c}_k)$ {`Get SAFER action`}
    $s_{k+1}, r_k, \omega_k \leftarrow$ Perform $\boldsymbol{a}_k$ in task $\mathcal{T}$
    Update $\pi_\theta(\boldsymbol{z}|\boldsymbol{s})$ using $(\boldsymbol{s}_k, \boldsymbol{z}_k, \boldsymbol{s}_{k+1}, r_k)$ to maximize $\mathcal{R}_\mu(\boldsymbol{s}_0)$
    Update Λ with $\boldsymbol{s}_k$ {`Update Λ in FIFO order`}
  end for
  Return: Policy $\pi_\theta(\boldsymbol{z}|\boldsymbol{s})$ for task $\mathcal{T}$

allowing for the inference of difficult-to-specify safety constraints through the task variable and discouraging unsafe behaviors. Finally, since SAFER can increase the likelihood of any safe behaviors, the final criteria that the offline primitive discovery technique can accelerate downstream policy learning will be met by using safe and successful trajectory data during SAFER training.

Parametization Choices To parameterize the SAFER action mapping f_ϕ, we use the Real NVP conditional normalizing flow, proposed by Dinh et al. (2017), due to it being highly expressive and allowing exact log-likelihood calculations. Next, we parameterize the posterior distribution $q_\rho(c|\Lambda)$ over the safety variable as a diagonal Gaussian. This choice allows computing the KL efficiently, while enabling an expressive task variable latent space. We use a transformer architecture to model the sequential dependency between Gaussian safety variable $\boldsymbol{c}$ and the window of previous states Λ (Vaswani et al., 2017). Finally, because the state space is an image pixel space, we also encode each observation to a vector using a CNN. An overview of the architecture is given in Figure 2.

Training It is necessary to use the reparameterization trick to compute gradients across the objective in Equation 5 (Kingma & Welling, 2014). Second, optimizing Equation 5 involves minimizing an unbounded log-likelihood in the second term of the objective. This term can lead to numerical instabilities when $p_\phi(\boldsymbol{a}_{\text{unsafe}}|\boldsymbol{s}, \boldsymbol{c})$ is too small. To overcome these issues, we use gradient clipping and freeze this term if it starts to diverge. Psuedo code of the procedure to train SAFER is provided in Appendix E in Algorithm 2 and hyperparameter details are provided in Appendix C. Last, note the objective requires access to unsafe data. We don't advise running an unsafe policy in the real world to collect unsafe data. Rather, our method should be applied in a domain where unsafe data already exists or where unsafe data can be collected in simulation (Srinivasan et al., 2020).

3.3. Accelerating Safe RL with SAFER

When using SAFER on a safe RL task, the goal is to accelerate safe learning by leveraging the mapping f_ϕ in the hierarchical policy $\mu_\psi(\boldsymbol{s}, \boldsymbol{c}) = \int_{\boldsymbol{z}} f_\phi(\boldsymbol{z}; \boldsymbol{s}; \boldsymbol{c}) d\pi_\theta(\boldsymbol{z}|\boldsymbol{s})$ where the policy parameters of the mapping ϕ are fixed and the parameters θ need to be optimized (Psuedo code of the procedure is provided in Algorithm 1). The policy $\pi_\theta(\boldsymbol{z}|\boldsymbol{s})$ can be learned by any standard RL methods (e.g., SAC (Haarnoja et al., 2018)) that produces continuous actions. To leverage SAFER at inference time, at each timestep t the RL policy takes an action in the abstract action space $\boldsymbol{z}_t \sim \pi_\theta(\boldsymbol{z}|\boldsymbol{s} = \boldsymbol{s}_t)$. Using the sliding window of states Λ, the safety variable posterior computes the distribution over the safety variable c_t.[2] Because a single safety variable value c_t is required, we fix it at its mean, $\mathrm{E}[\boldsymbol{c}_t] = \int \boldsymbol{c}\, dq_\rho(\boldsymbol{c}|\Lambda_t)$. Finally, SAFER computes the action $\boldsymbol{a}_t = f_\phi(\boldsymbol{z}_t; \boldsymbol{s}_t; \mathrm{E}[\boldsymbol{c}_t])$, the action is taken the environment, and the reward $r(\boldsymbol{s}_t, \boldsymbol{a}_t)$ and safety violations $\omega(\boldsymbol{s}_t, \boldsymbol{a}_t)$ are returned. The action $\boldsymbol{z}_t$ and reward r_t are added to the replay buffer for subsequent RL training.

3.4. Using SAFER to Guarantee Safety

Next, we demonstrate how it is straightforward to use SAFER to theoretically guarantee safety for *any* policy trained under the prior. To show this is the case, we first propose a condition about the environment, namely that there are always exist safe actions to take. We also ensure the SAFER prior is optimal according to (5):

Assumption 3.2. At every state $\boldsymbol{s}$, there always exists a safe action $\boldsymbol{a}$, i.e., $\forall \boldsymbol{s}\ \exists \boldsymbol{a}\ s.t.\ \omega(\boldsymbol{s}, \boldsymbol{a}) = 0$.

Assumption 3.3. The SAFER prior parameters $\hat{\rho}, \hat{\phi}$ are optimal per (5), such that all safe actions have higher likelihood than unsafe actions under the prior, i.e., $\forall \boldsymbol{a}, \boldsymbol{a}_{unsafe}, \boldsymbol{s}, \boldsymbol{c}$: $\log p_\phi(\boldsymbol{a}|\boldsymbol{s}, \boldsymbol{c}) \gg \log p_\phi(\boldsymbol{a}_{unsafe}|\boldsymbol{s}, \boldsymbol{c})$.

These assumptions establish that there are safe actions to take in the environment and that the SAFER prior has been trained in a way that optimizes its objective function. Then, we can construct a bound on the range of abstracts actions that ensures only safe actions under the prior:

Proposition 3.4. *There exists an η such that the corresponding bounded abstract actions $\boldsymbol{z} \in (-\eta, \eta)$ are safe, i.e., $\omega(\boldsymbol{s}, f_\phi(\boldsymbol{z}; \boldsymbol{c}; \boldsymbol{s})) = 0$, $\forall \boldsymbol{z} \in (-\eta, \eta), \boldsymbol{s}, \boldsymbol{c}$.*

According to this result, we can construct a latent variable bound around the mean of $\mathcal{Z}$ as the actions $f_\phi(\boldsymbol{z}; \boldsymbol{c}; \boldsymbol{s})$ that are more likely to be safe and successful are closer to the mean. Because unsafe actions under the SAFER prior have

[2] If there are insufficient states to compute a task window of size w (e.g., at the beginning of the rollout), we pad the available states with 0's in order to construct a window of w states.

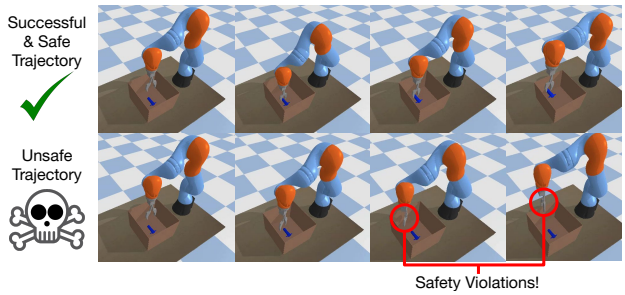

Figure 3. **An example of a task** where the robot successfully and safely grasping an object (top row). Here, the robot reaches into the container and extracts the object *without* touching the container. On the bottom row, the robot performs the same task but commits safety violations by touching the container.

much lower likelihood and Assumption 3.2 ensures that safe actions exists, there must exists a finite latent variable bound η that contains all safe actions. Finally, with such an η from Proposition 3.4, any policy π_θ that is trained under the SAFER prior and has a bounded abstraction action output $\boldsymbol{z} \in (-\eta, \eta)$ is safe, i.e., $\mathcal{W}_\mu(\boldsymbol{s}_0) = 0$. A more detailed proof is provided in Appendix A.

Therefore, we can use an offline data set with safe $(\boldsymbol{s}, \boldsymbol{a})$ and unsafe $(\boldsymbol{s}, \boldsymbol{a}_{\text{unsafe}})$ state-action pairs to determine the value of η that ensures safety in SAFER. To avoid having an overly tight bound, in practice it is acceptable to fix a range $(-\eta, \eta)$ that includes a small number of unsafe actions (e.g., at most $1 - \epsilon$ portion of all actions in data is unsafe).

Optimizing the real-valued $\eta > 0$ can be done by the following numerical gradient-free approach. First we initialize $\eta = \eta_0$ with a large enough constant and use that to generate the corresponding SAFER abstract actions $\boldsymbol{z}$ w.r.t. the offline data, whose latent variable is bounded in $(-\eta_0, \eta_0)$. We then sub-sample this SAFER-action bootstrapped dataset to construct a refined one that only has at most $1 - \epsilon$ portion of unsafe actions. Since the normalizing flow in SAFER is invertible, for every $(\boldsymbol{s}, \boldsymbol{a})$-pair in this data computing the corresponding latent action value is straightforward. This allows us to estimate η_1, which is the maximum latent action in this dataset. The above procedure is then repeated until convergence. Suppose the offline dataset contains sufficiently diverse state-action data that covers most situations encountered by SAFER. Then we would expect the above safety threshold to be rather generalizable and statistically significant (Kääriäinen & Langford, 2005). We call this procedure computing the SAFER *safety assurances* and provide pseudo code in Algorithm 3 in the Appendix.

4. Related Work

Safe Exploration A number of related works focus on safe exploration in RL when there is access to known constraint function (Wachi & Sui, 2020; Achiam et al., 2017; Dalal et al., 2018; Bharadhwaj et al., 2021; Narasimhan, 2020; Yang et al., 2021; Chow et al., 2018a;b; Achiam & Amodei, 2019; Berkenkamp et al., 2017; El Chamie et al., 2016; Turchetta et al., 2020). In our work, we focus on the setting where the constraint function cannot be easily specified and must be inferred from data, which is critical for scaling safe RL methods to the real world. To this end, a few works consider a similiar setting where the constraints must be inferred from data (Yang et al., 2017; Thananjeyan et al., 2021a). However, these works mainly focused on constrainted exploration and do not consider accelerating learning through learning useful behaviors within the space of safe $(\boldsymbol{s}, \boldsymbol{a})$ pairs, as is possible with SAFER.

Demonstrations for safe RL Using demonstrations to ensure safety in RL has received vast interests in the literature (Rosolia & Borrelli, 2018; Thananjeyan et al., 2020; 2021b; Driessens & Dzeroski, 2004; Smart & Kaelbling, 2000; Srinivasan et al., 2020). Though these works leverage demonstrations to improve safety, they do not handle data that are generated from different tasks with diverse skills.

Skill Discovery Various works consider learning skills in an online fashion (Eysenbach et al., 2019; Nachum et al., 2019; Sharma et al., 2020; Xie et al., 2021; Konidaris & Barto, 2009). These works learn skills that are used for planning (Sharma et al., 2020) or online RL (Eysenbach et al., 2019; Nachum et al., 2019). In contrast, we focus on a setting with access to an offline dataset, from which the primitives are learned. Further works also use offline datasets to extract skills, and transfer these to downstream learning (Pertsch et al., 2020; 2021; Ajay et al., 2021), but they do not consider safety.

Hierarchical RL Numerous works have found learning high level primitives using auxiliary models and controlling these with RL beneficial (Singh et al., 2021; Peng et al., 2019; Chandak et al., 2019; Nachum et al., 2018; Hausman et al., 2017; Florensa et al., 2017; Fox et al., 2017a; Dietterich, 1998; Rakelly et al., 2019). Though these works propose methods that are capable of accelerating the acquisition of successful policies, they do not specifically consider learning with safety constraints, which makes them susceptible to the generalization issues discussed in Section 2. In contrast, SAFER learns a hierarchical policy and also handles safety.

5. Experiments

We evaluate the calibration of the safety assurances introduced in Section 3.4 and how well SAFER encourages both safe and successful policy learning compared to baselines.

5.1. Experiments Setup

Recall that SAFER improves data efficiency by first learning safe primitives from offline data. To evaluate SAFER, we

Table 1. **Training RL with SAFER**, we give the mean ± SD success rate and cumulative safety violations across different tasks and initializations. SAFER produces the lowest cumulative safety violations throughout training and outperforms the baseline methods in terms of success rate. Methods without the use of offline primitive learning, namely SAC, are not able to learn during training (of 50,000 steps). These results demonstrate that SAFER is highly effective at encouraging safe learning.

	Success Rate (%)					
	Task 1	**Task 2**	**Task 3**	**Task 4**	**Task 5**	**Task 6**
SAC	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	2.3 ± 0.0
PARROT	0.0 ± 0.0	12.8 ± 0.2	25.7 ± 0.2	16.1 ± 0.2	33.9 ± 0.3	6.3 ± 0.1
Context PAR.	5.0 ± 0.0	24.2 ± 0.2	27.0 ± 0.3	0.7 ± 0.0	7.3 ± 0.1	12.0 ± 0.2
Prior Explore	1.8 ± 0.0	1.5 ± 0.0	3.0 ± 0.0	1.8 ± 0.0	1.1 ± 0.0	1.0 ± 0.0
SAFER	$\mathbf{21.0 \pm 0.1}$	$\mathbf{87.4 \pm 0.2}$	$\mathbf{89.3 \pm 0.0}$	$\mathbf{28.1 \pm 0.2}$	$\mathbf{54.4 \pm 0.1}$	$\mathbf{83.3 \pm 0.0}$
	Total Number of Safety Violations (Out of 50,000 Steps)					
	Task 1	**Task 2**	**Task 3**	**Task 4**	**Task 5**	**Task 6**
SAC	2045 ± 236	876 ± 117	1055 ± 216	2736 ± 147	2188 ± 405	756 ± 293
PARROT	6332 ± 3026	307 ± 291	13 ± 21	541 ± 461	2414 ± 314	932 ± 844
Context PAR.	5929 ± 2964	1576 ± 1208	1039 ± 777	5056 ± 1778	2796 ± 624	2085 ± 1951
Prior Explore	6203 ± 551	2240 ± 634	2867 ± 853	4525 ± 826	4669 ± 542	2596 ± 703
SAFER	$\mathbf{610 \pm 184}$	$\mathbf{51 \pm 61}$	$\mathbf{10 \pm 14}$	$\mathbf{455 \pm 470}$	$\mathbf{1707 \pm 292}$	$\mathbf{7 \pm 9}$

introduce a suite of safety-critical robotic grasping tasks that are inspired by the game Operation[3].

Safety-critical Robotic Grasping Tasks Inspired by the game Operation, whose primary goal is to extract objects from different containers without touching the container, we construct a set of 40 different grasping tasks, each consists of a container and object defined in PyBullet (Coumans & Bai, 2016–2021). We collect data from all these tasks to train SAFER but choose 6 of the more complex tasks for evaluation. In each of these tasks, the objects are randomly selected from ones available in PyBullet package, and the containers are generated to fit each object, whose dimensions (heights and widths) are generated randomly. Our agent controls the 5DoF robotic arm and gripper, for which a positive reward ($r(s, a) = 1$) is received when an object is extracted from the box, while a negative reward ($r(s, a) = -1$) is incurred at every time step before the task is complete. The agent incurs a safety violation ($\omega(s, a) = 1$) if the robotic arm touches the box (examples of safe/unsafe trajectories are in Figure 3, examples of the tasks are in Figure 10). The states are 48×48 pixel image observations of the scene collected from a fixed camera.

Offline Data Collection To generate the offline data for the SAFER training algorithm, for each robot grasping task we use the scripted policy from Singh et al. (2021) to collect trajectories with a total of $1,000,000$ steps. The scripted policy controls the robotic arm to grasp the object generally by minimizing the absolute distance between objects and the robot. To obtain more diverse/exploratory trajectories, one also adds random actuation noise to the policy. After collecting the trajectories, for each state-action pair $(\boldsymbol{s}, \boldsymbol{a})$ in the dataset we provide labels for **i)** safety violation $\omega(s, a) \in \{0, 1\}$, and **ii)** whether the pair $(\boldsymbol{s}, \boldsymbol{a})$ is part of a successful rollout (i.e., $(\boldsymbol{s}, \boldsymbol{a})$ such that $\mathbb{E}[r(\boldsymbol{s}_T, \boldsymbol{a}_T)|\mu_{\text{data}}, \boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \boldsymbol{a}] = 1$, where T is the trajectory length random variable). To create the state window Λ for SAFER training, for each $(\boldsymbol{s}, \boldsymbol{a})$ in the data buffer we save the previous w states. One can utilize these labels to categorize safe versus unsafe data to train SAFER.

Baseline Comparisons We compare against baseline methods that leverage demonstrations to accelerate learning, including PARROT (Singh et al., 2021), a contextual version of PARROT (Context. PAR) that uses a latent variable to help accelerate learning, Prior Explore (a method that samples from SAFER to help with data collection during training) and RL from scratch using SAC. See Appendix D for more details. We found RL from scratch methods did not rapidly solve our safe RL tasks. Thus, we did not compare against further RL from scratch methods, including safe RL techniques. Also, we assume we have trajectories labeled as safe and unsafe in our problem setting. We do not assume that we have access to an online safety constraint function that can tell us the safety value of a state during training, making comparisons to existing safe RL methods which use a constraint function during training difficult.

5.2. Results Discussion

Effectiveness of RL training with SAFER In Table 1 we compare SAFER with the baseline methods both in terms of cumulative safety violations and success rate. We choose a SAFER policy primitive with a safety assurance upper bound that guarantees at most 15% unsafe actions, which empirically maintains a good balance between performance

[3] https://en.wikipedia.org/wiki/OperationGame

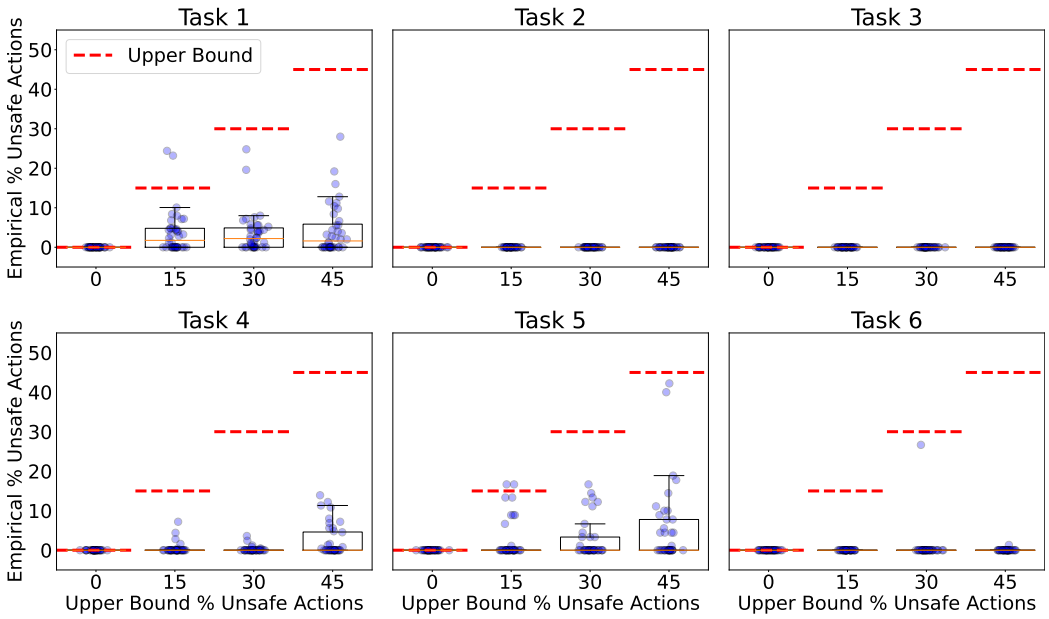

Figure 4. **Assessing the calibration of the SAFER safety assurances** by randomly sampling actions from the prior with various safety upper bounds across different evaluation tasks. Each dot corresponds to the empirical percent of unsafe $(\boldsymbol{s}, \boldsymbol{a})$ pairs from a single rollout on the task. Overall, we see that the SAFER safety assurances are quite well calibrated.

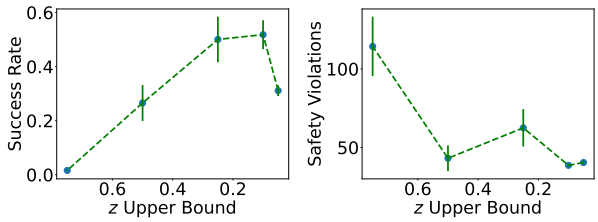

Figure 5. **Assessing the tradeoff between success and safety** varying the safety assurances bound on the abstract action space $\mathcal{Z}$, (referred to as η in Algorithm 3). There is an sweet spot where success rate is high and safety violations is low.

and safety. For each downstream task, we then train the RL agent π_θ with SAC for only $50,000$ steps because we are more interested to evaluate the power of the primitive learning algorithm. Overall, we see that SAFER has the lowest cumulative safety violations, indicating that it is the most effective method in promoting safe policy learning. Interestingly, SAFER also consistently outperforms other methods in policy performance. The strong success rates of SAFER are potentially due to the fact that discouraging unsafe behaviors may indeed help refining the space of useful behaviors, thus improves policy learning.

Safety Assurance Calibration We evaluate whether the safe abstract action bound of SAFER computed in Section 3.4 is well calibrated, i.e., the empirical percent of unsafe actions should be less than the upper bound. To study this, we compute the $\mathcal{Z}$-action bound $(-\eta, \eta)$ corresponding to an upper bound of 0%, 15%, 30% and 45% unsafe actions for SAFER. We compute the percentage of unsafe actions by randomly sampling actions from SAFER on each evaluation task and report the results in Figure 4, showing that the SAFER bounds are indeed well calibrated.

Success & Safety Tradeoff In Figure 5 we assess the tradeoff between success and safety by varying the $\mathcal{Z}$-action bound in Algorithm 3. We sweep over different bounds and compute both the success rate and safety violations at the end of training for Task 5. We see that there is a sweet spot with high success rate and low safety violations when the safety assurances bound is close to 15%. Interestingly when the bound is too tight (corresponding small $\boldsymbol{z}$ values), both the safety violation and success rate become low, indicating SAFER cannot solve the task without sufficient exploration.

Impact of latent safety variable We train SAFER on Tasks 2 and 5 using the contrastive objective in Equation 5 but without the safety variable. In this case, the success rate never exceeds 10% and the safety violations are quite high (see Appendix B for the Task 2 results). In contrast, the safety variable in SAFER has at least a 60% success rate on both tasks (Table 1). This result suggests that the latent safety variable is crucial for success and safety.

6. Conclusion

In this paper, we introduced SAFER, an offline primitive learning method that improves the data efficiency of safe RL

when there is access to both safe and unsafe data examples. This is particularly important because most existing safe RL algorithms are very data inefficient. We proposed a set of complex safety-critical robotic grasping tasks to evaluate SAFER, investigated limitations of state-of-the-art offline primitive learning baselines, and demonstrated that SAFER can achieve better success rates while enforcing safety with high-probability assurances.

References

Achiam, J. and Amodei, D. Benchmarking safe exploration in deep reinforcement learning. 2019.

Achiam, J., Held, D., Tamar, A., and Abbeel, P. Constrained policy optimization. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 22–31. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/achiam17a.html.

Ajay, A., Kumar, A., Agrawal, P., Levine, S., and Nachum, O. Opal: Offline primitive discovery for accelerating offline reinforcement learning. *ICLR*, abs/2010.13611, 2021.

Altman, E. Constrained markov decision processes, 1999.

Berkenkamp, F., Turchetta, M., Schoellig, A., and Krause, A. Safe model-based reinforcement learning with stability guarantees. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/766ebcd59621e305170616ba3d3dac32-Paper.pdf.

Bharadhwaj, H., Kumar, A., Rhinehart, N., Levine, S., Shkurti, F., and Garg, A. Conservative safety critics for exploration. *ICLR*, 2021.

Chandak, Y., Theocharous, G., Kostas, J., Jordan, S., and Thomas, P. Learning action representations for reinforcement learning. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 941–950. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/chandak19a.html.

Charnes, A. and Cooper, W. W. Chance-constrained programming. *Management science*, 6(1):73–79, 1959.

Chow, Y., Nachum, O., Duenez-Guzman, E., and Ghavamzadeh, M. A lyapunov-based approach to safe reinforcement learning. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, pp. 8103–8112, Red Hook, NY, USA, 2018a. Curran Associates Inc.

Chow, Y., Nachum, O., Duenez-Guzman, E., and Ghavamzadeh, M. A lyapunov-based approach to safe reinforcement learning. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018b. URL https://proceedings.neurips.cc/paper/2018/file/4fe5149039b52765bde64beb9f674940-Paper.pdf.

Coumans, E. and Bai, Y. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2021.

Dalal, G., Dvijotham, K., Vecerík, M., Hester, T., Paduraru, C., and Tassa, Y. Safe exploration in continuous action spaces. *CoRR*, abs/1801.08757, 2018. URL http://arxiv.org/abs/1801.08757.

Dietterich, T. G. The maxq method for hierarchical reinforcement learning. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, pp. 118–126, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc. ISBN 1558605568.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using real NVP. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=HkpbnH9lx.

Driessens, K. and Dzeroski, S. Integrating guidance into relational reinforcement learning. *Machine Learning*, 57: 271–304, 2004.

Dulac-Arnold, G., Evans, R., van Hasselt, H., Sunehag, P., Lillicrap, T., Hunt, J., Mann, T., Weber, T., Degris, T., and Coppin, B. Deep reinforcement learning in large discrete action spaces. *arXiv preprint arXiv:1512.07679*, 2015.

El Chamie, M., Yu, Y., and Açıkmeşe, B. Convex synthesis of randomized policies for controlled markov chains with density safety upper bound constraints. In *2016 American Control Conference (ACC)*, pp. 6290–6295, 2016. doi: 10.1109/ACC.2016.7526658.

Eysenbach, B., Gupta, A., Ibarz, J., and Levine, S. Diversity is all you need: Learning skills without a reward function. *ICLR*, 2019.

Fetaya, E., Jacobsen, J.-H., Grathwohl, W., and Zemel, R. Understanding the limitations of conditional generative models. *ICLR*, 10 2020.

Florensa, C., Duan, Y., and Abbeel, P. Stochastic neural networks for hierarchical reinforcement learning. *ICLR*, 2017.

Fox, R., Krishnan, S., Stoica, I., and Goldberg, K. Multi-level discovery of deep options. *ArXiv*, abs/1703.08294, 2017a.

Fox, R., Krishnan, S., Stoica, I., and Goldberg, K. Multi-level discovery of deep options. *arXiv preprint arXiv:1703.08294*, 2017b.

García, J., Fern, and o Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(42):1437–1480, 2015. URL http://jmlr.org/papers/v16/garcia15a.html.

Ghadirzadeh, A., Poklukar, P., Kyrki, V., Kragic, D., and Björkman, M. Data-efficient visuomotor policy training using reinforcement learning and generative models. *arXiv preprint arXiv:2007.13134*, 2020.

Guadarrama, S., Korattikara, A., Ramirez, O., Castro, P., Holly, E., Fishman, S., Wang, K., Gonina, E., Wu, N., Kokiopoulou, E., Sbaiz, L., Smith, J., Bartók, G., Berent, J., Harris, C., Vanhoucke, V., and Brevdo, E. TF-Agents: A library for reinforcement learning in tensorflow. https://github.com/tensorflow/agents, 2018. URL https://github.com/tensorflow/agents. [Online; accessed 25-June-2019].

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *ICML*, 2018.

Hausman, K., Chebotar, Y., Schaal, S., Sukhatme, G., and Lim, J. J. Multi-modal imitation learning from unstructured demonstrations using generative adversarial nets. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 1235–1245, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Kääriäinen, M. and Langford, J. A comparison of tight generalization error bounds. In *Proceedings of the 22nd international conference on Machine learning*, pp. 409–416, 2005.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *ICLR*, 2014.

Kirichenko, P., Izmailov, P., and Wilson, A. G. Why normalizing flows fail to detect out-of-distribution data. *arXiv*, 2020.

Konidaris, G. and Barto, A. Skill discovery in continuous reinforcement learning domains using skill chaining. In Bengio, Y., Schuurmans, D., Lafferty, J., Williams, C., and Culotta, A. (eds.), *Advances in Neural Information Processing Systems*, volume 22. Curran Associates, Inc., 2009. URL https://proceedings.neurips.cc/paper/2009/file/e0cf1f47118daebc5b16269099ad7347-Paper.pdf.

Nachum, O., Gu, S. S., Lee, H., and Levine, S. Data-efficient hierarchical reinforcement learning. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/e6384711491713d29bc63fc5eeb5ba4f-Paper.pdf.

Nachum, O., Gu, S., Lee, H., and Levine, S. Near-optimal representation learning for hierarchical reinforcement learning. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=H1emus0qF7.

Nalisnick, E., Matsukawa, A., Teh, Y., Gorur, D., and Lakshminarayanan, B. Do deep generative models know what they don't know? *ICLR*, 10 2018.

Narasimhan, K. Projection-based constrained policy optimization. *ICLR*, abs/2010.03152, 2020.

Nemirovski, A. and Shapiro, A. Convex approximations of chance constrained programs. *SIAM Journal on Optimization*, 17(4):969–996, 2007. doi: 10.1137/050622328. URL https://doi.org/10.1137/050622328.

Peng, X. B., Chang, M., Zhang, G., Abbeel, P., and Levine, S. *MCP: Learning Composable Hierarchical Control with Multiplicative Compositional Policies*. Curran Associates Inc., Red Hook, NY, USA, 2019.

Pertsch, K., Lee, Y., and Lim, J. J. Accelerating reinforcement learning with learned skill priors. *CoRL*, 2020.

Pertsch, K., Lee, Y., Wu, Y., and Lim, J. J. Guided reinforcement learning with learned skills. In *Self-Supervision for Reinforcement Learning Workshop-ICLR 2021*, 2021.

Rakelly, K., Zhou, A., Finn, C., Levine, S., and Quillen, D. Efficient off-policy meta-reinforcement

learning via probabilistic context variables. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5331–5340. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/rakelly19a.html.

Rosolia, U. and Borrelli, F. Learning model predictive control for iterative tasks. a data-driven control framework. *IEEE Transactions on Automatic Control*, 63:1883–1896, 2018.

Shankar, T. and Gupta, A. Learning robot skills with temporal variational inference. In *International Conference on Machine Learning*, pp. 8624–8633. PMLR, 2020.

Sharma, A., Gu, S. S., Levine, S., Kumar, V., and Hausman, K. Dynamics-aware unsupervised discovery of skills. *ArXiv*, abs/1907.01657, 2020.

Singh, A., Liu, H., Zhou, G., Yu, A., Rhinehart, N., and Levine, S. Parrot: Data-driven behavioral priors for reinforcement learning. *ICLR*, 2021.

Smart, W. D. and Kaelbling, L. P. Practical reinforcement learning in continuous spaces. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pp. 903–910, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc. ISBN 1558607072.

Srinivasan, K. P., Eysenbach, B., Ha, S., Tan, J., and Finn, C. Learning to be safe: Deep rl with a safety critic. *ArXiv*, abs/2010.14603, 2020.

Thananjeyan, B., Balakrishna, A., Rosolia, U., Li, F., McAllister, R., Gonzalez, J., Levine, S., Borrelli, F., and Goldberg, K. Safety augmented value estimation from demonstrations (saved): Safe deep model-based rl for sparse cost robotic tasks. *IEEE Robotics and Automation Letters*, 5: 3612–3619, 2020.

Thananjeyan, B., Balakrishna, A., Nair, S., Luo, M., Srinivasan, K. P., Hwang, M., Gonzalez, J. E., Ibarz, J., Finn, C., and Goldberg, K. Recovery rl: Safe reinforcement learning with learned recovery zones. *IEEE Robotics and Automation Letters*, 6:4915–4922, 2021a.

Thananjeyan, B., Balakrishna, A., Rosolia, U., Gonzalez, J., Ames, A. D., and Goldberg, K. Abc-lmpc: Safe sample-based learning mpc for stochastic nonlinear dynamical systems with adjustable boundary conditions. In *WAFR*, 2021b.

Turchetta, M., Kolobov, A., Shah, S., Krause, A., and Agarwal, A. Safe reinforcement learning via curriculum induction. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M. F., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 12151–12162. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/8df6a65941e4c9da40a4fb899de65c55-Paper.pdf.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Wachi, A. and Sui, Y. Safe reinforcement learning in constrained Markov decision processes. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 9797–9806. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/wachi20a.html.

Xie, K., Bharadhwaj, H., Hafner, D., Garg, A., and Shkurti, F. Latent skill planning for exploration and transfer. *ICLR*, 2021.

Yang, T., Hu, M., Chow, Y., Ramadge, P. J., and Narasimhan, K. Safe reinforcement learning with natural language constraints. *CoRR*, abs/2010.05150, 2017. URL https://arxiv.org/abs/2010.05150.

Yang, T.-Y., Rosca, J. P., Narasimhan, K., and Ramadge, P. J. Accelerating safe reinforcement learning with constraint-mismatched policies. *ICML*, abs/2006.11645, 2021.

Zhang, C., Bütepage, J., Kjellström, H., and Mandt, S. Advances in variational inference. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41:2008–2026, 2019.

Appendix

A. Guaranteeing Safety with SAFER

In this section, we demonstrate how it is straightforward to show SAFER can guarantee safety for any policy trained under the skill prior, demonstrating the utility of the method. We restate and clarify our assumptions before providing the proof of the proposition. The first assumption ensures that there is always a safe action to take.

Assumption A.1. At every state $\boldsymbol{s}$, there always exists a safe action $\boldsymbol{a}$, i.e., $\forall \boldsymbol{s}\ \exists \boldsymbol{a}\ s.t.\ \omega(\boldsymbol{s}, \boldsymbol{a}) = 0$.

The second assumption ensures that the SAFER model is optimal according to the SAFER objective given in Objective 5. In effect, this assumption means that safe actions have high likelihood while the unsafe actions are much less likely under the SAFER prior.

Assumption A.2. The SAFER prior parameters $\hat{\rho}, \hat{\phi}$ are optimal per Objective 5, such that all safe actions have higher likelihood than unsafe actions under the prior, i.e., $\forall \boldsymbol{a}, \boldsymbol{a}_{unsafe}, \boldsymbol{s}, \boldsymbol{c}:\ \log p_\phi(\boldsymbol{a}|\boldsymbol{s}, \boldsymbol{c}) \gg \log p_\phi(\boldsymbol{a}_{unsafe}|\boldsymbol{s}, \boldsymbol{c})$.

Next, we provide a proof for Proposition 3.4.

Proposition A.3. *There exists an η such that the corresponding bounded abstract actions $\boldsymbol{z} \in (-\eta, \eta)$ are safe, i.e., $\omega(\boldsymbol{s}, f_\phi(\boldsymbol{z}; \boldsymbol{c}; \boldsymbol{s})) = 0, \forall \boldsymbol{z} \in (-\eta, \eta), \boldsymbol{s}, \boldsymbol{c}$.*

Proof (Sketch): Because the abstract action space $\mathcal{Z}$ is unit Gaussian ($\mathcal{Z} \sim \mathcal{N}(0, I)$) it is the case that the $\boldsymbol{z}$'s that are closer to the zero vector $\mathbf{0}$ have higher likelihood, i.e., $\boldsymbol{z}$'s with lower norm $||\boldsymbol{z}||$ have higher likelihood. From the assumption, we know that in every state there exists safe actions and with the way we train SAFER it will have much higher likelihood than unsafe actions in the prior distribution, $\log p_\phi(\boldsymbol{a}|\boldsymbol{s}, \boldsymbol{c}) \gg \log p_\phi(\boldsymbol{a}_{unsafe}|\boldsymbol{s}, \boldsymbol{c})$. Using the invertibility property of normalizing flow, one concludes that for all state and action pairs, the unsafe abstract actions $\boldsymbol{z}$ are much farther away from the zero vector $\mathbf{0}$. Consequently, there must exist a finite latent bound η that separates all safe actions with unsafe ones. □

B. Additional Results

In this Appendix, we present additional results with SAFER.

Cumulative Safety Violation Graphs In the main paper, we presented the cumulative safety violations at the end of training. Here, we present graphs of the cumulative safety violations in figure 6 throughout training for the baselines and SAFER. In these graphs, we see that SAFER is consistently the safety method throughout training.

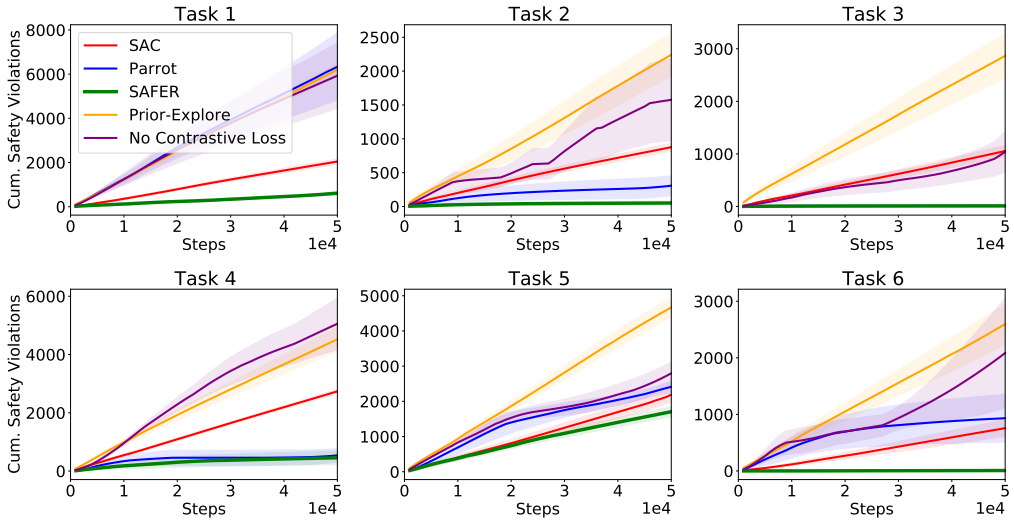

Figure 6. **The cumulative safety violations throughout training** for SAFER and the baselines. We see that SAFER is consistently the safest method throughout training.

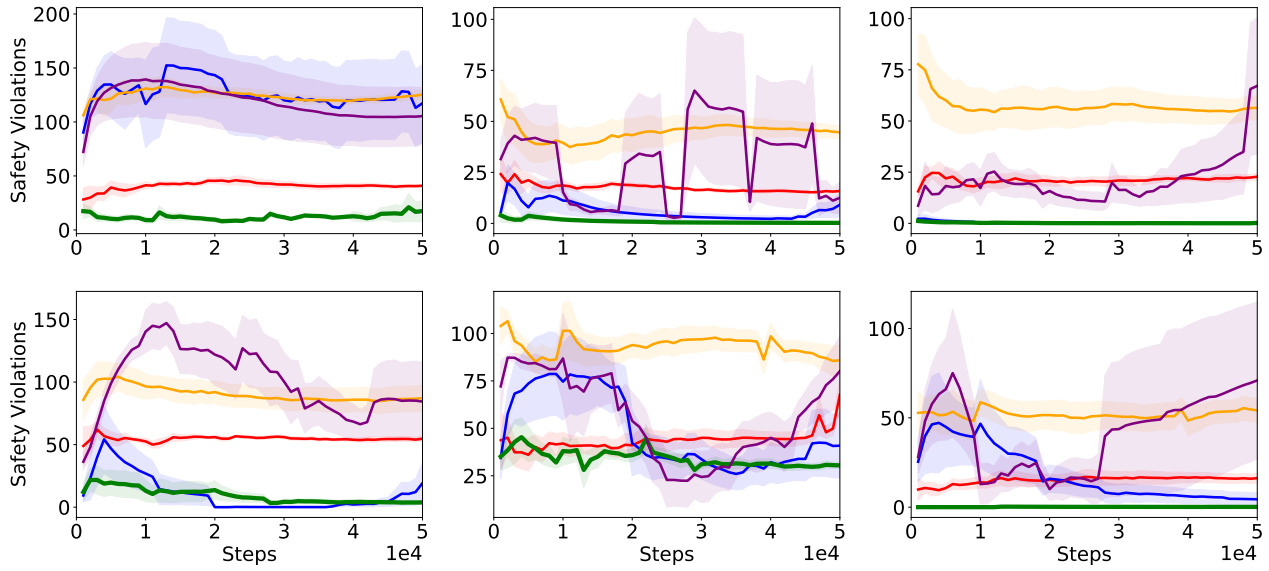

Figure 7. The safety violations over each step of training for each of the tasks (same task ordering as Figure 6). We see that SAFER is consistently the most safe method throughout training.

Per Step Safety Violations In the main paper, we provide *cumulative* safety violation graphs. Here, we provide the safety violations over the last $1,000$ steps in Figure 7 in order to get a better sense of the safety violations throughout training. We again consistently see SAFER is safety method over the course of training. One interesting observation is that, in Section 2, we discussed how PARROT rates unsafe $(\boldsymbol{s}, \boldsymbol{a})$ pairs as high likelihood. Because PARROT draws on higher likelihood actions from the prior earlier in training, we would expect that PARROT would be more unsafe earlier in training. Empirically, we see this to be the case. Looking at the graphs, PARROT has high safety violation spikes at the beginning of training. These results demonstrate that our earlier observations surrounding the unsafety of PARROT hold true when running RL.

Impact of Probabilistic Treatment One question worth considering is how necessary is it to treat SAFER as a latent variable model and optimize the posterior over the safety variable using variational inference, as is proposed in Section 3.2. It could be easier to treat $\boldsymbol{c}$ as a vector (without defining it as a Guassian random variable), exclude the KL term from Equation 5, and optimize $q_\rho(\boldsymbol{c}|\Lambda)$ with the rest of the objective. To assess whether this is the case, we ran a sweep across different hyperparameter configurations, including the number of bijectors in the real NVP model, the learning rate, λ, and the number of hidden units in each bijector. Doing this, however, we find SAFER quickly diverges, indicating the probabilistic treatment greatly helps stabilize training and is necessary for the success of the method.

Training SAFER Without the Safety Context Variable As an abalation in the main paper, we considered training SAFER without the SAFETY context variable and found that it led to worse success rate and relatively higher safety violations. In this Appendix, we provide the full training results in Figure 8 in terms of success rate and per step safety violations. Here, we see that for the tasks considered, training SAFER without the safety variable leads to worse success rates and less safety (compared to the per step success rates in Figure 7).

Training PARROT With Unsafe Data In the paper, we performed experiments PARROT trained using safe data. Meaning, $w(\boldsymbol{s}, \boldsymbol{a}) = 0$ for each training point. We also limited the data to only those tuples in successful trajectories to promote PARROT acquiring safe and successful behaviors. Though it makes the most sense to train PARROT for safety concerned tasks in this fashion, it is worth considering what would happen if we also included unsafe data from successful trajectories. To assess what would happen, we train PARROT using both safe and unsafe data from successful trajectories, using the hyperparameters for PARROT in Section C. The results given in Figure 9 demonstrate that this leads to relatively higher per step safety violations, indicating that it is best to train PARROT with *only* safe data from successful trajectories.

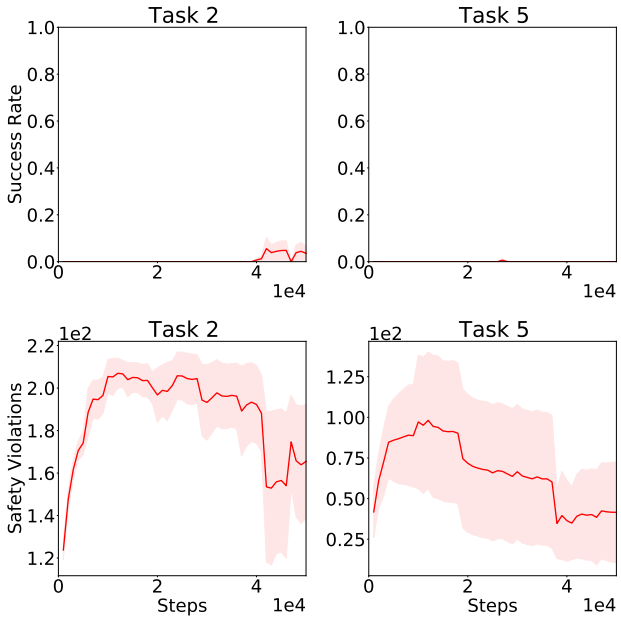

Figure 8. Effectiveness of RL Training using the SAFER objective *without* the safety variable. We see the prior without the safety variable is quite unsuccessful, indicating that the safety variable is critical to enabling SAFER to promote both safe and successful learning.

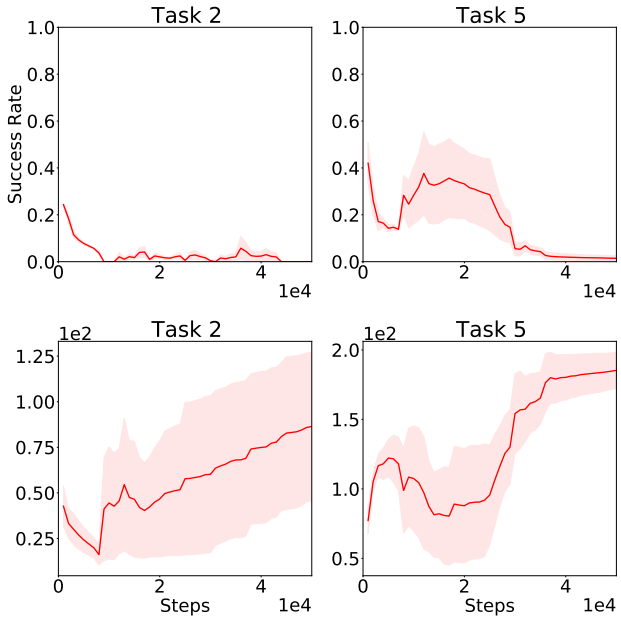

Figure 9. **Training PARROT using unsafe data** from successful trajectories as well as safe data. We see that this leads to leads to relatively worse success rates (top row) as well as relatively higher per step safety violations (bottom row). These results suggest it is best to train PARROT with safe and successful data only.

C. SAFER Hyperparameter Details

Hyperparameter Details We explored a number of different parameter configurations with SAFER. We tuned λ $(1e-4, 1e-5)$, the number of bijectors in the real NVP flow model $(3, 5)$, the number of components in the context variable $\boldsymbol{c}$ $(8, 32, 64)$, the size of the states window w $(16, 32)$, the optimizer (Adam, SGD+Momentum), and the learning rate $(1e-4, 5e-5)$. We trained for 500k steps and found that using a smaller number of components in the context variable led to more stable training (8). Setting the learning rate to $(1e-4)$ led to much quicker convergence, without sacrificing much stability. Furthermore, training with Adam led to divergence in some cases while SGD+Momentum tended to diverge less often. Between the other parameters considered, there was relatively little difference, and therefore we used a model with learning rate $1e-4$, 3 bijectors, 8 components, 16 states window size, and SGD+Momentum.

D. Baseline Methods

We select several baseline methods to compare with SAFER. We mainly focus on methods leverage action primitives trained with offline data to improve efficiency, e.g., PARROT, Prior-Explore. While we are aware of additional baseline methods, e.g., TrajRL (Shankar & Gupta, 2020; Fox et al., 2017b), HIRL (Ghadirzadeh et al., 2020), in the literature, we omit their comparisons here because it has been shown in prior work (Singh et al., 2021) that their performance is consistently below that of the state of the art.

Soft Actor Critic: Soft-actor critic (SAC) (Haarnoja et al., 2018) is one of the standard model-free policy-gradient based RL methods. Here without using any action primitives we apply SAC to learn a policy that directly maps states in $\mathcal{X}$ to actions in $\mathcal{A}$. Later we also use SAC in all our action primitive based RL methods (e.g., SAFER, PARROT) to optimize the high-level policy. Therefore, one can view the SAC baseline as one ablation study as well. We use the implementation from TF-Agents (Guadarrama et al., 2018). We used SAC with autonmatic entropy tuning and tune the number of target network update period, discount factor, policy learning rate, and Q-function learning rate.

PARROT: We compare against the state-of-the-art primitive learning RL method PARROT, proposed by Singh et al. (2021). Similar to SAFER, PARROT leverages a conditional normalizing flow and to train a behavioral prior using data from successful rollouts. To enforce safety in the PARROT agent, we additionally limit the training data of its behavioral prior to *both* safe and successful rollouts, otherwise PARROT may encourage unsafe behaviors. We tune the number of bijectors in the conditional normalizing flow for PARROT (5, 3), the number hidden units in each bijector layer (128, 256), the learning rate ($1e-4$, $5e-5$, $1e-5$), the optimizer (Adam or SGD+Momentum), and train for 500k steps. We find using 3 bijectors with learning rate $1e-4$, and the Adam optimizer works best.

Prior-Explore: We also consider the prior-explore method proposed in Singh et al. (2021) as one of our baseline method. Here the prior-explore policy combines the mapping f_ϕ action policy in Equation 2 with an SAC agent to aid exploration of the RL agent. It selects an action from the prior policy with probability δ and from the SAC agent otherwise. Followed from Singh et al. (2021), we set this probability δ to 0.9 and use mapping f_ϕ trained for SAFER.

Contextual PARROT (SAFER Without Contrastive Loss): As one ablation study we consider SAFER *without* the contrastive loss. This setup also models the behavioral prior policy with a conditional normalizing flow and the latent safety variable but trains that only with safe and successful data. Note that this baseline method is equivalent to PARROT, with a policy that is a function of the latent safety variable. We use the same parameters as PARROT with this baseline and 8 components in safety variable because we found this number of components to be the most successful with SAFER.

E. Training SAFER

In this appendix, we provide psuedo code for the SAFER training procedure in Algorithm 2.

Algorithm 2 SAFER Training

Require: SAFER Behavioral Prior f_ϕ, Safety Variable Posterior $q_\rho(\boldsymbol{c}|\Lambda)$, safe dataset $\mathcal{D}_{\text{safe}}$, unsafe dataset $\mathcal{D}_{\text{unsafe}}$, Steps N, λ

Let `flow_loss(·)` refer to Equation 2

for $n = 1, ..., N$ **do**

  $(\boldsymbol{s}, \boldsymbol{a}, \Lambda)_{\text{Safe}} \sim \mathcal{D}_{\text{Safe}}$ {Sample safe + unsafe batches of data}

  $(\boldsymbol{s}, \boldsymbol{a}, \Lambda)_{\text{Unsafe}} \sim \mathcal{D}_{\text{Unsafe}}$

  $\boldsymbol{c}_{\text{Safe}} \sim q_\rho(c|\Lambda_{\text{Safe}})$ {Sample safety variables}

  $\boldsymbol{c}_{\text{Unsafe}} \sim q_\rho(c|\Lambda_{\text{Safe}})$

  $\mathcal{L}_{\text{safe}} \leftarrow \log\left(\texttt{flow_loss}(\boldsymbol{s}_{\text{safe}}; \boldsymbol{a}_{\text{safe}}; \boldsymbol{c}_{\text{safe}})\right)$ {Compute log-likelihoods}

  $\mathcal{L}_{\text{unsafe}} \leftarrow \log\left(\texttt{flow_loss}(\boldsymbol{s}_{\text{unsafe}}; \boldsymbol{a}_{\text{unsafe}}; \boldsymbol{c}_{\text{unsafe}})\right)$

  $D_{\text{KL}}^{\text{Safe}} \leftarrow D_{\text{KL}}\left(q_\rho(\boldsymbol{c}|\Lambda_{\text{Safe}})||p(\boldsymbol{c})\right)$ {Compute KL of safety variables}

  $D_{\text{KL}}^{\text{Unsafe}} \leftarrow D_{\text{KL}}\left(q_\rho(\boldsymbol{c}|\Lambda_{\text{Unsafe}})||p(\boldsymbol{c})\right)$

  $NLL \leftarrow -(\mathcal{L}_{\text{safe}} - \lambda \cdot \mathcal{L}_{\text{unsafe}} - D_{\text{KL}}^{\text{Safe}} - D_{\text{KL}}^{\text{Unsafe}})$

  Minimize NLL and update ϕ, ρ {Update SAFER}

end for

Return: SAFER Behaviors Prior f_ϕ, Safety Variable Posterior $q_\rho(\boldsymbol{c}|\Lambda)$

F. Setting the Safety Assurance

In this appendix, we provide psuedo code for the SAFER safety assurances procedure in Algorithm 3. This algorithm provided a numerical gradient-free approach to find an optimal bound η that included ϵ portion safe actions.

Algorithm 3 SAFER Safety Assurances

Require: Initial bound η_0, Desired percent safe actions ϵ, SAFER prior f_ϕ, Safe dataset $\mathcal{D}_{\text{safe}}$, Unsafe dataset $\mathcal{D}_{\text{unsafe}}$

define

function get_in_bound(dataset $\mathcal{D}$, bound η_t)

  `// This function computes the abstract actions` $\boldsymbol{z}$ `within bound` η_t

  $\mathcal{Z} \leftarrow \{\}$

  for $(\boldsymbol{s}, \boldsymbol{a}, \Lambda)$ in $\mathcal{D}$ **do**

    `// Iterate over tuple (state` $\boldsymbol{s}$`, action` $\boldsymbol{a}$`, and context` Λ`)`

    $\boldsymbol{c} \leftarrow \mathbb{E}_{q_\rho(\cdot|\Lambda)}[\boldsymbol{c}]$, $\boldsymbol{z} \leftarrow f_\phi^{-1}(\boldsymbol{a}; \boldsymbol{s}; \boldsymbol{c})$ {Get abstract action $\boldsymbol{z}$ from $\boldsymbol{s}$, $\boldsymbol{a}$, and Λ}

    if $\boldsymbol{z}$ within bound η_t **then**

      $\mathcal{Z} = \mathcal{Z} \cup \boldsymbol{z}$ {Add $\boldsymbol{z}$ if its within bound η_t}

    end if

  end for

  return $\mathcal{Z}$

end function

$\eta \leftarrow (-\eta_0, \eta_0)$ {Initialize bound}

`done` $\leftarrow$ False

while not `done` **do**

  $\mathcal{Z}_{\text{safe}}^{\eta}$ = get_in_bound($\mathcal{D}_{\text{safe}}$, η), $\mathcal{Z}_{\text{unsafe}}^{\eta}$ = get_in_bound($\mathcal{D}_{\text{unsafe}}$, η) {Get $\boldsymbol{z}$ in current bound η}

  $S \leftarrow |\mathcal{Z}_{\text{safe}}^{\eta}| + |\mathcal{Z}_{\text{unsafe}}^{\eta}|$

  $\mathcal{Z}_{\text{safe}}^{\epsilon} \sim$ sample $\lfloor S \times \epsilon \rfloor$ items from $\mathcal{Z}_{\text{safe}}^{\eta}$, $\mathcal{Z}_{\text{unsafe}}^{1-\epsilon} \sim$ sample $\lfloor S \times (1-\epsilon) \rfloor$ items from $\mathcal{Z}_{\text{unsafe}}^{\eta}$

  $\eta \leftarrow$ the max component absolute value across $\mathcal{Z}_{\text{unsafe}}^{1-\epsilon}$ and $\mathcal{Z}_{\text{safe}}^{\epsilon}$ {Update bound}

  if ϵ portion of items across $\mathcal{D}_{\text{safe}}$ and $\mathcal{D}_{\text{unsafe}}$ within bound $(-\eta, \eta)$ are safe **then**

    `done` $\leftarrow$ True {Break if bound η contains desired portion safe actions}

  end if

end while

return η

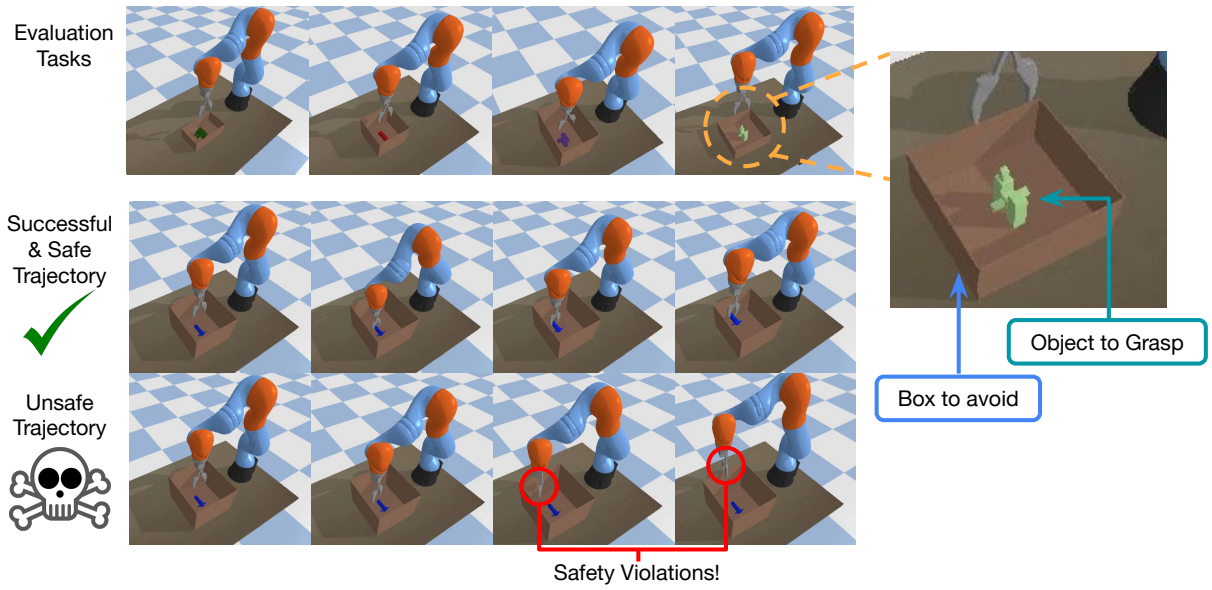

Figure 10. Additional examples of tasks included in the safe robotic grasping environment (top row). The tasks all use different sizes containers, to represent different difficulties in preserving safe behavior. We also provide a zoomed in version of the task (right hand side). Finally, we also include the examples of safe and unsafe trajectories provided in the main paper (Figure 3) for completeness

G. Additional Task Examples

In this Appendix, we provide additional examples of the tasks included in the safe robotic grasping environment in Figure 10.