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

Dylan Slack[*]
UC Irvine
dslack@uci.edu

Yinlam Chow
Google Research
yinlamchow@google.com

Bo Dai
Google Research
bodai@google.com

Nevan Wichers
Google Research
wichersn@google.com

Abstract: Methods that extract policy primitives from offline demonstrations using deep generative models have shown promise at accelerating reinforcement learning (RL) for new tasks. Intuitively, these methods should also help to train *safe* RL agents because they enforce useful skills. However, we identify these techniques are not well equipped for safe policy learning because they ignore negative experiences (e.g., unsafe or unsuccessful), focusing only on positive experiences, which harms their ability to generalize to new tasks safely. Rather, we model the latent *safety context* using principled contrastive training on an offline dataset of demonstrations from many tasks, including both negative and positive experiences. Using this latent variable, our RL framework, SAFEty skill pRiors (SAFER) extracts task specific safe primitive skills to safely and successfully generalize to new tasks. 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,[2] in which SAFER outperforms state-of-the-art primitive learning methods in success and safety.

Keywords: Primitives, Offline RL, Behavioral Prior

1 Introduction

Reinforcement learning (RL) has demonstrated strong 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), extensive interaction with an environment is impossible due to dangers associated with exploration. These difficulties are further complicated by the challenging nature of specifying safety constraints in complex environments. Nevertheless, relatively few existing safe reinforcement learning algorithms can *rapidly* and *safely* solve complex RL problems with hard to specify safety constraints.

One promising route is offline primitive learning methods [1, 2, 3, 4]. These methods use offline datasets to learn representations of useful actions or *behaviors* through deep 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 the generative model learns useful primitives from a dataset, simplifying the action space [5].

[*]Work performed while an intern at Google AI

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

Decision Awareness in Reinforcement Learning Workshop at the 39th International Conference on Machine Learning (ICML), Baltimore, Maryland, USA, 2022. Copyright 2022 by the author(s).

However, primitive learning techniques suffer from a critical drawback when applied to safety concerned tasks. 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, when offline skill learning methods 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 increases the likelihood of unsafe actions (see Fig. 1) [6, 7, 8]. *Thus, current state of the art primitive learning techniques may, counter-intuitively, encourage unsafe behavior*.

In this work, we identify that modeling the latent *safety context* is the key to overcoming these challenges. To this end, we introduce *SAFER*: safety skill priors, a primitive learning technique that *accelerates* reinforcement learning with *safe* actions. (An overview is provided in Figure 2.) To acquire safe skills, SAFER **i)** uses a contrastive loss to distinguish safe and unsafe data and **ii)** learns a posterior sampling distribution of a latent safety variable, that captures different safety contexts. Using the safety context, SAFER established a set of task specific safe actions, greatly improving safety generalization. Consequently, policies trained using the SAFER abstract actions as the action space will learn to compose a set of safe policy primitives. 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 state of the art primitive learning methods, SAFER has both a higher success rate and fewer safety violations.

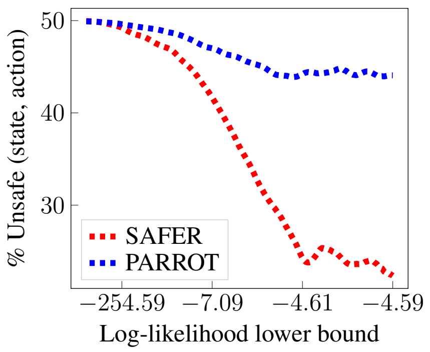

Figure 1: **Evaluating the concentration of unsafe data in high likelihood regions** by computing the % of unsafe state-action pairs in a holdout dataset of a safe robotic grasping task. PARROT assigns high likelihoods to unsafe data, i.e., it does not encourage safety, while SAFER has much lower likelihood in unsafe data, so it will encourage safety.

2 Related Work

Safe Exploration Several related works focus on safe exploration in RL when there is access to known constraint function [9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]. In our work, we focus on the setting where the constraint function cannot be easily specified and must be inferred *entirely* 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. Thananjeyan et al. [21] uses an offline dataset of safety constraint violations to learn about safety constraints and trains a policy to recover from safety violations so the agent can continue exploring safely. Yang et al. [22] use natural language to enforce a set of safety constraints during policy learning. Compared to our work, these methods focus on constrained exploration in a single task setting. Instead, we consider accelerating learning across multiple tasks through learned safe primitives.

Demonstrations for Safe RL The use of demonstrations to ensure safety in RL has received considerable interest in the literature [23, 24, 25, 26]. Most relevant, Srinivasan et al. [27] use unsafe demonstrations to constrain exploration to only a safe set of actions for task adaption. Thananjeyan et al. [28] relies on a set of sub-optimal demonstrations to safely learn new tasks. Though these works leverage demonstrations to improve safety, they each rely on task specific demonstrations. Instead, we focus on learning generalizable safe primitives, which we transfer to downstream tasks, and we demonstrate this can greatly accelerate safe policy learning.

Skill Discovery Various works consider learning skills in an online fashion [29, 30, 31, 32, 33]. These methods learn skills for planning [31] or online RL [29, 30]. In contrast, we focus on a setting with access to an offline dataset, from which the primitives are learned. Further works also use offline

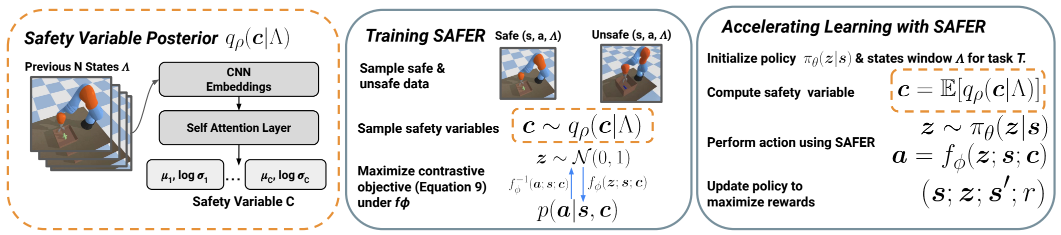
Safety Variable Posterior $q_\rho(\boldsymbol{c}|\Lambda)$
Previous N States Λ
CNN Embeddings
Self Attention Layer
μ_1, log σ_1 … μ_C, log σ_C
Safety Variable C
Training SAFER
Safe (s, a, Λ)
Unsafe (s, a, Λ)
Sample safe & unsafe data
Sample safety variables
$\boldsymbol{c} \sim q_\rho(\boldsymbol{c}|\Lambda)$
$\boldsymbol{z} \sim \mathcal{N}(0,1)$
Maximize contrastive objective (Equation 9) under $f\phi$
$f_\phi^{-1}(\boldsymbol{a};\boldsymbol{s};\boldsymbol{c})$
$f_\phi(\boldsymbol{z};\boldsymbol{s};\boldsymbol{c})$
$p(\boldsymbol{a}|\boldsymbol{s},\boldsymbol{c})$
Accelerating Learning with SAFER
Initialize policy $\pi_\theta(z|s)$ & states window Λ for task T.
Compute safety variable
$\boldsymbol{c} = \mathbb{E}[q_\rho(\boldsymbol{c}|\Lambda)]$
Perform action using SAFER
$\boldsymbol{z} \sim \pi_\theta(\boldsymbol{z}|\boldsymbol{s})$
$\boldsymbol{a} = f_\phi(\boldsymbol{z};\boldsymbol{s};\boldsymbol{c})$
Update policy to maximize rewards
$(\boldsymbol{s};\boldsymbol{z};\boldsymbol{s}';r)$

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).

datasets to extract skills, and transfer these to downstream learning [2, 3, 4], but they do not model the safety of the downstream tasks, which we demonstrate is critical for safe generalization.

Hierarchical RL Numerous works have found learning high level primitives using auxiliary models and controlling these with RL beneficial [1, 34, 35, 36, 37, 38, 39, 40, 41]. 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 1 and Section 3, where these methods can inadvertently make unsafe behavior high likelihood. In contrast, SAFER learns a hierarchical policy that explicitly considers the safety of tasks, resulting in both safe and successful generalization to downstream tasks, addressing the aforementioned issues.

3 Background

In this section, we provide background for our problem setting. Recall the motivating household robotics example where we wish to train an agent to accomplish a series of household tasks. The agent must learn to do tasks like set a cast iron pot to boil, remove dirt off a dish with a sponge, or cut an apple with a knife. Within these tasks, there are different goals and notions of safety. For example, the robot can safely drop the sponge but cannot safely drop the cast iron pot while cooking, because this would be quite dangerous. From a training perspective, it is difficult to devise safety violation functions for all tasks, given how many ways one could behave unsafely with a cast iron pot or knife. However, it is straightforward to determine whether the task is successful (e.g., the apple is cut in half or it isn't). Consequently, it is more reasonable to assume an *offline data collection process* where a large set of behaviors have been annotated for success and safety violation, through simulation or real world experience, and agents must rely entirely on the existing data to learn safety constraints when generalizing to downstream tasks, though they may have access to a sparse reward signal.

Safety MDP In a setting with different tasks and safety constraints, for each task $\mathcal{T}$, the agent's interaction is modeled as a safety Markov decision process decision process (safety MDP). A safety MDP is a tuple $(\mathcal{S}, \mathcal{A}, \mathrm{T}, r, \gamma, \boldsymbol{s}_0, \omega(\boldsymbol{s}, \boldsymbol{a}))$, where $\mathcal{S}$ 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, $\boldsymbol{s}_0 \in \mathcal{S}$ is the initial state, and $\omega(\boldsymbol{s}, \boldsymbol{a}) \in \{0, 1\}$ is the safety violation function, that indicates whether the current state and action lead to a safety violation (1) or no safety violation (0). Given a policy μ, 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 satisfy the safety constraint while maximizing expected return.

Offline Primitive Learning In many problems, we may not have access to the underlying reward and safety violation functions across many different tasks. Instead, we assume access to 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, where the reward and safety violation are labeled for each state action pair. Further, when adapting to *new tasks*, we assume that we *do not* have access to the underlying safety violation function and only have a sparse reward signal for whether the task was completed successfully. Thus, the safety constraints must be inferred entirely from the data.

To use the offline dataset $\mathcal{D}$ to generalize to downstream tasks, offline *primitive discovery* techniques [1, 4, 2, 3], use a policy structure consisting of a *prior* $\mu_\psi = f_\phi(\boldsymbol{z}; \boldsymbol{s})$ and *policy* $\boldsymbol{z} \sim \pi_\theta(\boldsymbol{z}|\boldsymbol{s})$. In this parameterization, the prior $f_\phi : \mathcal{Z} \times \mathcal{S} \to \mathcal{A}$ with learnable parameters ϕ maps from the abstract action space $\mathcal{Z}$ and state space $\mathcal{S}$ to the action space $\mathcal{A}$ and is trained to learn a set of useful skills from the dataset $\mathcal{D}$. 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 $\mathcal{Z}$. In this way, policies ϕ_θ trained on downstream tasks learn to compose the primitives learned by f_ϕ from the offline dataset. Different ways to express the behavior prior mapping have been proposed and have been found to greatly accelerate policy learning. For instance, Ajay et al. [4] 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. [1] directly optimizes the log-likelihood, log $p(\boldsymbol{a}|\boldsymbol{s})$, and fix an invertible mapping through the use of a conditional normalizing flow [42] between the abstract action space $\mathcal{Z}$ and the distribution over useful actions $p(\boldsymbol{a}|\boldsymbol{s})$.

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 [1, 4]. 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 [6, 7, 8]. 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. Since the proposed offline primitive discovery policy structure relies on high likelihood actions from the prior [4, 1], using the aforementioned behavior priors for safety will be problematic.

4 SAFER: Safety Skill Priors

Considering the shortcomings mentioned in Section 3 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.

4.1 Latent Safety Variable

To address these criteria, we a latent variable called the *safety variable* $\boldsymbol{c} \in \mathcal{C}$ that encodes *safety context* about the environment, i.e., $f_\phi : \mathcal{Z} \times \mathcal{C} \times \mathcal{S} \to \mathcal{A}$. This construction encodes information beyond the current state $\boldsymbol{s}$ to help SAFER model complex per task safety dynamics. For example, the safety variable could encode the locations of people or animals while a robot performs household tasks. Because we do not assume the task variable $\mathcal{C}$ is provided, we infer it from a network.

4.2 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 [42] 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

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

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 [43] 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 [44]. 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. 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. We write the evidence-lower bound (ELBO) of our model as

$$\mathbb{E}_{\boldsymbol{c}\sim q_\rho(\cdot|\Lambda)}\left[\log p_\phi(\boldsymbol{a}|\boldsymbol{s},\boldsymbol{c})\right] - D_{\text{KL}}(q_\rho(\cdot|\Lambda)||p(\cdot)), \tag{2}$$

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 [45] to the ELBO optimization,

$$\begin{aligned}\max_{\rho,\phi,\xi} \mathbb{E}_{\boldsymbol{c}\sim q_\rho(\cdot|\boldsymbol{s})}\left[\log p_\phi(\boldsymbol{a}|\boldsymbol{s},\boldsymbol{c})\right] - D_{\text{KL}}\left(q_\rho(\cdot|\Lambda)||p(\cdot)\right) - \lambda'\xi \\ \text{s.t. } \mathbb{P}_{\boldsymbol{c}\sim q_\rho(\cdot|\boldsymbol{s})}(p_\phi(\boldsymbol{a}_{\text{unsafe}}|\boldsymbol{s},\boldsymbol{c}) > \epsilon) \leq \xi,\end{aligned} \tag{3}$$

where the constraint states that 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 makes 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 [45]. We provide a proof in Appendix Section A.

Proposition 4.1 *Assuming the chance constrained ELBO is written as in Equation 3, we can write the surrogate lower bound as,*

$$\max_{\rho,\phi} \mathbb{E}_{\boldsymbol{c}\sim q_\rho(\cdot|\boldsymbol{s})}\Big[\log p_\phi(\boldsymbol{a}|\boldsymbol{s},\boldsymbol{c}) - \lambda \log p_\phi(\boldsymbol{a}_{unsafe}|\boldsymbol{s},\boldsymbol{c})\Big] - D_{KL}(q_\rho(\cdot|\Lambda)||p(\cdot)) \tag{4}$$

We denote this objective as the *SAFER Contrastive Objective*. Further, this objective function has an intuitive interpretation. The first two terms act as a contrastive loss that *encourages* safe actions (high likelihood) while *discourages* unsafe ones (low likelihood). Together with the final term, the variable $\boldsymbol{c}$ is forced to contain useful information about safety. Thus the objective satisfies our goals, allowing for the inference of 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. [42], 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 to compute the KL efficiently while enabling an expressive 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 [46]. 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.

Algorithm 1 Accelerating Safe Reinforcement Learning with SAFER

```
Require: SAFER Prior $f_\phi$, Safety Posterior $q_\rho(\boldsymbol{c}|\Lambda)$,
  Safety bound $\eta$, 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}
    $\boldsymbol{s}_{k+1}, r_k, \omega_k \leftarrow$ Perform $\boldsymbol{a}_k$ in task $\mathcal{T}$
    Update $\pi_\theta(\boldsymbol{z}|\boldsymbol{s})$ using $(\boldsymbol{s}_k, \boldsymbol{z}_k, \boldsymbol{s}_{k+1}, r_k)$
    Update $\Lambda$ with $\boldsymbol{s}_k$ in FIFO order
  end for
  Return: Policy $\pi_\theta(\boldsymbol{z}|\boldsymbol{s})$ for task $\mathcal{T}$
```

Training It is necessary to use the reparameterization trick to compute gradients across the objective in Equation 5 [47]. 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 F in Algorithm 2 and hyperparameter details are provided in Appendix D.

4.3 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 [48]) 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.[3] Because a single safety variable value c_t is required, we fix it at its mean, $\text{E}[\boldsymbol{c}_t] = \int \boldsymbol{c}\, dq_\rho(\boldsymbol{c}|\Lambda_t)$. Finally, SAFER computes the action $\boldsymbol{a}_t = f_\phi(\boldsymbol{z}_t; \boldsymbol{s}_t; \text{E}[\boldsymbol{c}_t])$, 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.

4.4 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 assume there always exists safe actions to take in the environment and make an optimiality assumptions about the prior in (5). Then, we can construct a bound on the range of abstracts actions that ensures only safe actions under the prior:

Proposition 4.2 *There exists an η such that the corresponding bounded abstract actions $\boldsymbol{z} \in (-\eta, \eta)$ are safe, i.e., $\omega(\boldsymbol{s}, f_\phi(\boldsymbol{z}; \boldsymbol{c}; \boldsymbol{s})) = 0$, $\forall \boldsymbol{z} \in (-\eta, \eta), \boldsymbol{s}, \boldsymbol{c}$.*

As a 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 lower likelihood and our assumption ensures that safe actions exists, there must exist a finite latent variable bound η that contains all safe actions. Consequently, with such an η from Proposition 4.2, any agent π_θ that is trained under the SAFER prior and has a bounded abstraction action output $\boldsymbol{z} \in (-\eta, \eta)$ is safe. The full proof details are provided in Appendix B.

In practice, we 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. Also, 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) to avoid an overly tight bound. We optimize the real-valued $\eta > 0$ by a numerical gradient-free approach. First, we initialize $\eta = \eta_0$ with a large 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 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. We repeat this procedure until convergence. If the offline dataset contains sufficiently diverse state-action data that covers most situations encountered by SAFER, we expect the above safety threshold to be generalizable [49]. We denote this procedure computing the SAFER *safety assurances* and provide pseudo code in Algorithm 3 in the Appendix.

5 Experiments

We evaluate the calibration of the safety assurances introduced in Section 4.4 and how well SAFER encourages both safe and successful policy learning compared to baselines.

5.1 Experiments Setup

To evaluate SAFER, we introduce a suite of safety-critical robotic grasping tasks that are inspired by the game Operation[4].

Safety-critical Robotic Grasping Tasks Based on the game Operation, whose goal is to extract objects from different sized containers without touching the container, we construct a set of 40

[3]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.

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

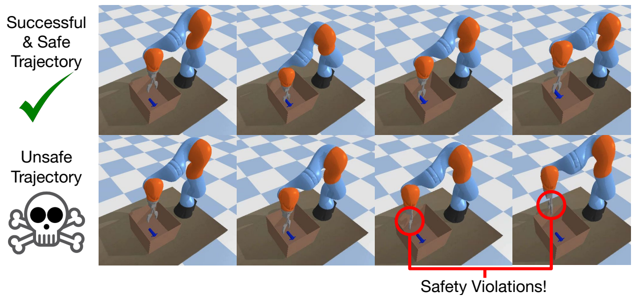

Figure 3: **An example of a task** where the robot successfully and safely grasping an object (top row). Here, the robot reaches into the container and extracts the object *without* touching the container. On the bottom row, the robot performs the same task but commits safety violations by touching the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

grasping tasks, each consisting of a container and object defined in PyBullet [50]. We collect data from these tasks to train SAFER and use 6 of the more complex tasks for evaluation. In our tasks, the objects are randomly selected from ones available in PyBullet package, and the containers are generated to fit the objects, whose dimensions (heights and widths) are generated randomly. Our agent controls a 5DoF robotic arm and gripper. The agent receives positive reward ($r(s, a) = 1$) when it extracts the object from the box and a negative reward ($r(s, a) = -1$) at every time step while the task is incomplete. The agent incurs a safety violation ($\omega(s, a) = 1$) if the arm touches the box (examples of safe/unsafe trajectories 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. [1] 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 To demonstrate the improved safety performance of SAFER over existing offline primitive learning techniques, we compare against baseline methods that leverage offline data to accelerate learning, including PARROT [1], 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 E for more details. Last, because our setting requires learning safety constraints *entirely* from labeled offline data, we do not compare against methods that require online safety constraint functions, such as many existing safe RL methods which use a constraint function during training.

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. Note, here we use the underlying reward and safety violation functions for each task to evaluate performance. 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 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 4.4 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

Table 1: **Training RL with SAFER**, we give the mean $\pm$ 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.

	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	$\mathbf{610 \pm 184}$	$\mathbf{51 \pm 61}$	$\mathbf{10 \pm 14}$	$\mathbf{455 \pm 470}$	$\mathbf{1707 \pm 292}$	$\mathbf{7 \pm 9}$

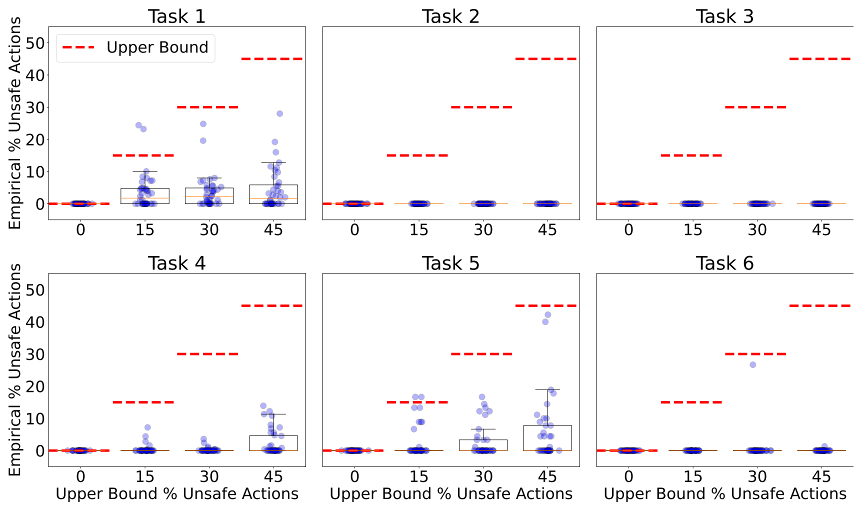

Figure 4: **Assessing the calibration of the SAFER safety assurances** by randomly sampling actions from the prior with various safety upper bounds across different evaluation tasks. Each dot corresponds to the empirical percent of unsafe $(\boldsymbol{s}, \boldsymbol{a})$ pairs from a single rollout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

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.

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 C 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 Limitations

Though SAFER improves both safe and successful generalization to downstream tasks, there are several critical limitations to consider. Foremost, SAFER relies on a labeled offline dataset. In certain settings, it may be impractical to collect a sufficiently large dataset to ensure useful learned primitives or to receive high quality labels. If the dataset is not sufficiently large or the labels are poor quality, this could harm the capacity of SAFER to learn useful primitives. In the future, researchers should benchmark and improve the sample efficiency of SAFER. Second, the offline dataset includes a selection of unsafe demonstrations. In settings where there are not existing unsafe data points (e.g., from previous failures), or unsafe data cannot be simulated, it may be difficult for SAFER to learn generalizable safety constraints from the data.

7 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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[17] J. Achiam and D. Amodei. Benchmarking safe exploration in deep reinforcement learning. 2019.

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

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

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

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

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

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

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

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

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

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

[28] B. Thananjeyan, A. Balakrishna, U. Rosolia, F. Li, R. McAllister, J. Gonzalez, S. Levine, F. Borrelli, and K. Goldberg. 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.

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

[30] O. Nachum, S. Gu, H. Lee, and S. Levine. 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.

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

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

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

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

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

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

[37] K. Hausman, Y. Chebotar, S. Schaal, G. Sukhatme, and J. J. Lim. 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, page 1235–1245, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

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

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

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

[41] K. Rakelly, A. Zhou, C. Finn, S. Levine, and D. Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5331–5340. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/rakelly19a.html.

[42] L. Dinh, J. Sohl-Dickstein, and S. Bengio. 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.

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

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

[45] A. Nemirovski and A. Shapiro. 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.

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

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

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

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

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

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

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

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

[54] S. Guadarrama, A. Korattikara, O. Ramirez, P. Castro, E. Holly, S. Fishman, K. Wang, E. Gonina, N. Wu, E. Kokiopoulou, L. Sbaiz, J. Smith, G. Bartók, J. Berent, C. Harris, V. Vanhoucke, and E. Brevdo. 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].

Appendix

A Proof: Tractable Lower Bound

Proposition 3.1: Assuming the chance constrained ELBO is written as in Equation 3, we can write the surrogate lower bound as,

$$\max_{\rho,\phi} \mathbb{E}_{\boldsymbol{c}\sim q_\rho(\cdot|\boldsymbol{s})}\Big[logp_\phi(\boldsymbol{a}|\boldsymbol{s},\boldsymbol{c})-\lambda logp_\phi(\boldsymbol{a}_{\text{unsafe}}|\boldsymbol{s},\boldsymbol{c})\Big] - D_{\text{KL}}(q_\rho(\cdot|\Lambda)||p(\cdot)) \tag{5}$$

Proof: We rewrite the optimization 3 into the following form,

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

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 3:

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

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.

B 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 B.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 B.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., $\forall \boldsymbol{a},\boldsymbol{a}_{unsafe},\boldsymbol{s},\boldsymbol{c}$: $logp_\phi(\boldsymbol{a}|\boldsymbol{s},\boldsymbol{c}) \gg logp_\phi(\boldsymbol{a}_{unsafe}|\boldsymbol{s},\boldsymbol{c})$.*

Next, we provide a proof for Proposition 4.2.

Proposition B.1 *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. ■

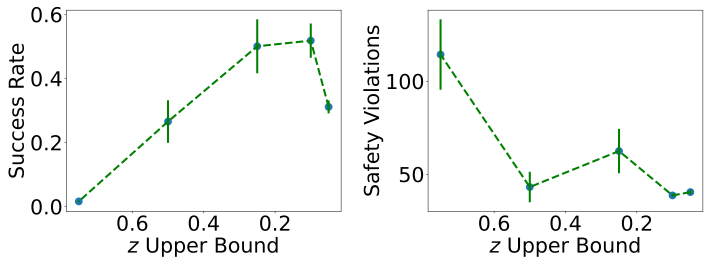

Figure 6: **Assessing the tradeoff between success and safety** varying the safety assurances bound on the abstract action space $\mathcal{Z}$, (referred to as η in Algorithm 3). There is an sweet spot where success rate is high and safety violations is low.

C 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 5 throughout training for the baselines and SAFER. In these graphs, we see that SAFER is consistently the safety method throughout training.

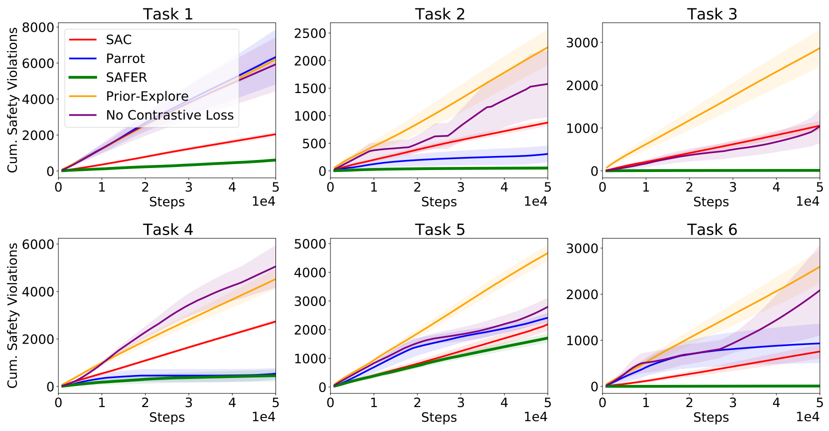

Figure 5: **The cumulative safety violations throughout training** for SAFER and the baselines. We see that SAFER is consistently the safest method throughout training.

Success & Safety Tradeoff In Figure 6 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.

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 3, 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.

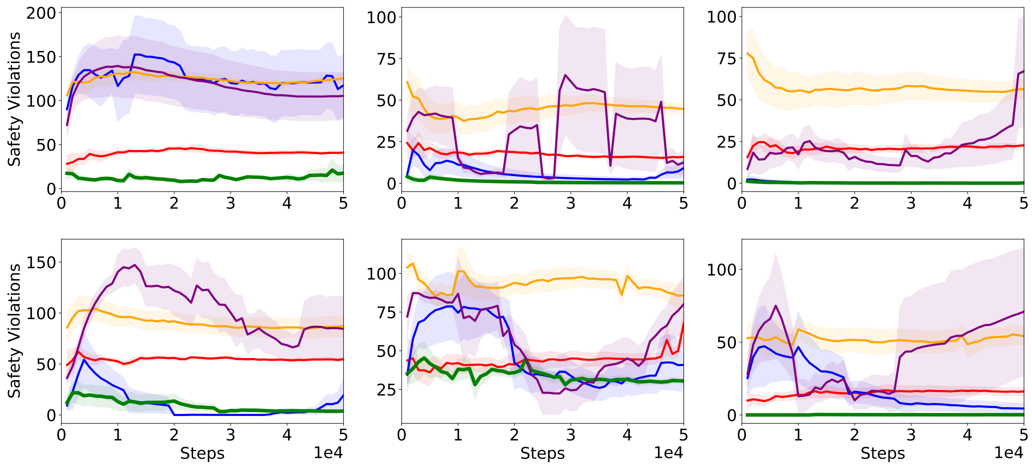

Figure 7: The safety violations over each step of training for each of the tasks (same task ordering as Figure 5). We see that SAFER is consistently the most safe method throughout training.

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 4.2. 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 ablation 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 D. 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.

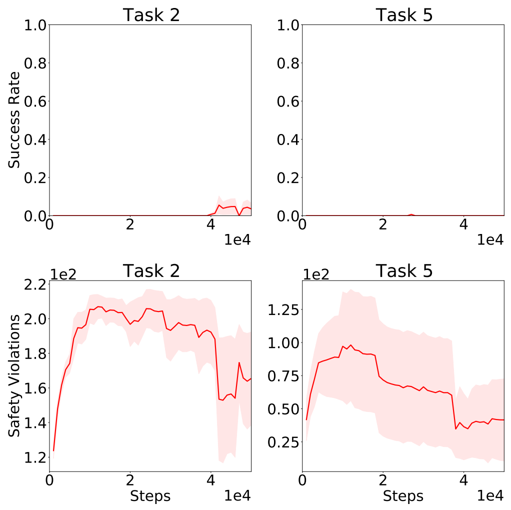

Figure 8: Effectiveness of RL Training using the SAFER objective *without* the safety variable. We see the prior without the safety variable is quite unsuccessful, indicating that the safety variable is critical to enabling SAFER to promote both safe and successful learning.

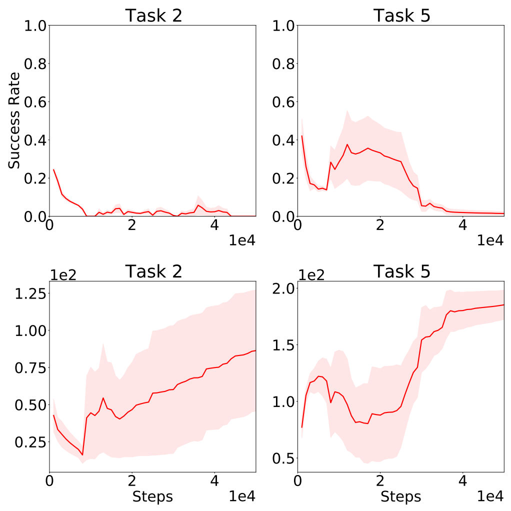

Figure 9: **Training PARROT using unsafe data** from successful trajectories as well as safe data. We see that this leads to leads to relatively worse success rates (top row) as well as relatively higher per step safety violations (bottom row). These results suggest it is best to train PARROT with safe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

D 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.

E 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 [51, 52], HIRL [53], in the literature, we omit their comparisons here because it has been shown in prior work [1] that their performance is consistently below that of the state of the art.

Soft Actor Critic: Soft-actor critic (SAC) [48] 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 [54]. 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. [1]. 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. [1] as one of our baseline method. Here the prior-explore policy combines the mapping f_ϕ action policy in Equation 1 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. [1], 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.

F 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 $f_\phi$, Safety Variable Posterior $q_\rho(\boldsymbol{c}|\Lambda)$, safe dataset $\mathcal{D}_{\text{safe}}$, unsafe
  dataset $\mathcal{D}_{\text{unsafe}}$, Steps $N$, $\lambda$
  Let flow_loss(·) refer to Equation 1
  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 $\phi, \rho$                                    {Update SAFER}
  end for
  Return: SAFER Behaviors Prior $f_\phi$, Safety Variable Posterior $q_\rho(\boldsymbol{c}|\Lambda)$
```

G 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 $\eta_0$, Desired percent safe actions $\epsilon$, SAFER prior $f_\phi$, Safe dataset $\mathcal{D}_{\text{safe}}$, Unsafe
  dataset $\mathcal{D}_{\text{unsafe}}$
  define
  function get_in_bound(dataset $\mathcal{D}$, bound $\eta_t$)
    // This function computes the abstract actions $\boldsymbol{z}$ within bound $\eta_t$
    $\mathcal{Z} \leftarrow \{\}$
    for $(\boldsymbol{s}, \boldsymbol{a}, \Lambda)$ in $\mathcal{D}$ do
      // Iterate over tuple (state $\boldsymbol{s}$, action $\boldsymbol{a}$, and context $\Lambda$)
      $\boldsymbol{c} \leftarrow \mathbb{E}_{q_\rho(\cdot|\Lambda)}[\boldsymbol{c}]$, $\boldsymbol{z} \leftarrow f_\phi^{-1}(\boldsymbol{a}; \boldsymbol{s}; \boldsymbol{c})$   {Get abstract action $\boldsymbol{z}$ from $\boldsymbol{s}$, $\boldsymbol{a}$, and $\Lambda$}
      if $\boldsymbol{z}$ within bound $\eta_t$ then
        $\mathcal{Z} = \mathcal{Z} \cup \boldsymbol{z}$                          {Add $\boldsymbol{z}$ if its within bound $\eta_t$}
      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($\mathcal{D}_{\text{safe}}$, $\eta$), $\mathcal{Z}_{\text{unsafe}}^{\eta}$ = get_in_bound($\mathcal{D}_{\text{unsafe}}$, $\eta$) {Get $\boldsymbol{z}$ in current bound
    $\eta$}
    $S \leftarrow |\mathcal{Z}_{\text{safe}}^{\eta}| + |\mathcal{Z}_{\text{unsafe}}^{\eta}|$
    $\mathcal{Z}_{\text{safe}}^{\epsilon} \sim$ 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 $\epsilon$ 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 $\eta$ contains desired portion safe actions}
    end if
  end while
  return $\eta$
```

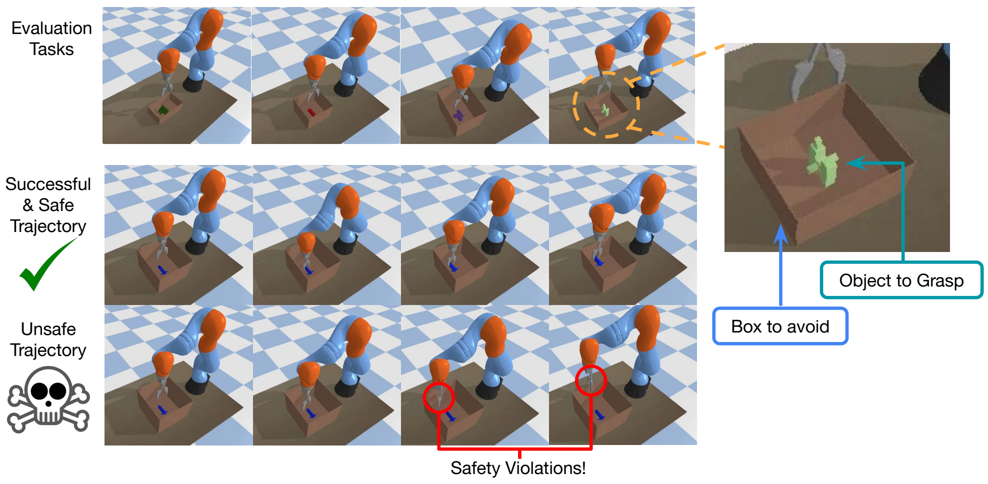

Figure 10: Additional examples of tasks included in the safe robotic grasping environment (top row). The tasks all use different sizes containers, to represent different difficulties in preserving safe behavior. We also provide a zoomed in version of the task (right hand side). Finally, we also include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

H Additional Task Examples

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