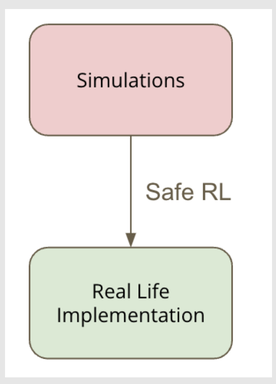
Human in the Loop, Safe Reinforcement Learning for Continuous Control

Chetan Reddy N [1] Arvind Ragghav V [1] Nirav Bhatt [2]

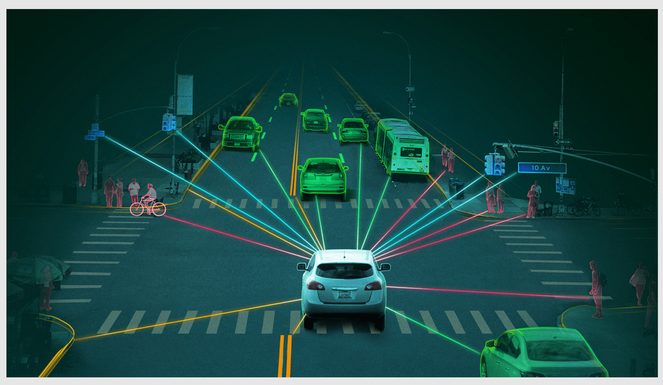
[1]Robert Bosch Centre for Data Science and AI (RBCDSAI), Indian Institute of Technology, Madras [2]Department of Data Science and AI, Wadhwani School of Data Science and AI, Department of Biotechnology, Indian Institute of Technology Madras

Introduction

- One of the biggest obstacles for Reinforcement Learning to moving from simulated environments to real-world applications is safety.
- RL agents learn new tasks in uncertain environments through extensive exploration and trial and error, which can sometimes violate safety constraints and result in undesirable outcomes.
- Safe RL encapsulates algorithms that deal with this tradeoff between exploration and safety. [Kazantzidis et al. 2022] [Brunke et al. 2022]
- In this study, we define a human-in-the-loop framework that ensures safety in both training and deployment.



(a) 'Safe' RL as the bridge



(b) Autonomous Driving

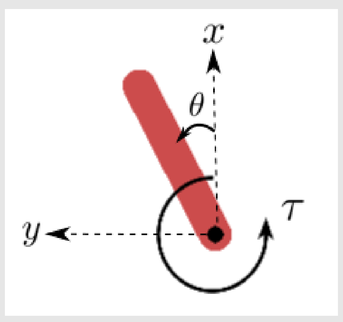
(c) Healthcare Robotics

Figure 1. Motivation

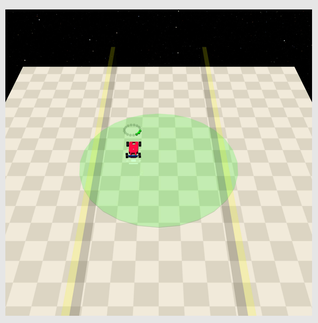
Data and Environments

We focus on continuous control environments which have a wide range of applications in robotics and autonomous systems. To train and evaluate our algorithm, we have used the following libraries with benchmark environments:

- OpenAI Gym - Inverted Pendulum Environment
- Safety Gymnasium - PointCircle [Ji et al. 2023]



(a) Inverted Pendulum Environment



(b) Safety Gymnasium

Figure 2. Environments

Methodology

We make some definitions before explaining the methodology:

- $\mathcal{X}_s$: Truly Safe State Space
- $\mathcal{X}_{ms}$: Marginally Safe State Space
- SSS: Safe State Space $\mathcal{X}_s \cup \mathcal{X}_{ms}$
- CSP or π_{safe}: Conservative Safe Policy

Our approach involves two steps:

- The first step is the utilization of expert human input to establish a safe state space (SSS) and a corresponding conservative safe policy (CSP). In simple environments, the human directly provides SSS and CSP while in complex environments, safe trajectories generated by the human are used to estimate SSS and CSP.

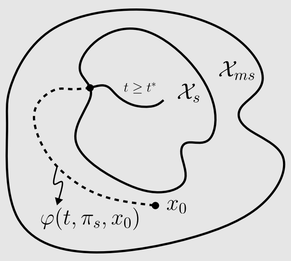


Figure 3. SSS and CSP Formulation

- In the second step, we use a modified version of the Deep Deterministic Policy Gradient algorithm augmented with a safety layer (built using SSS and CSA) that is learned by the agent during the training.

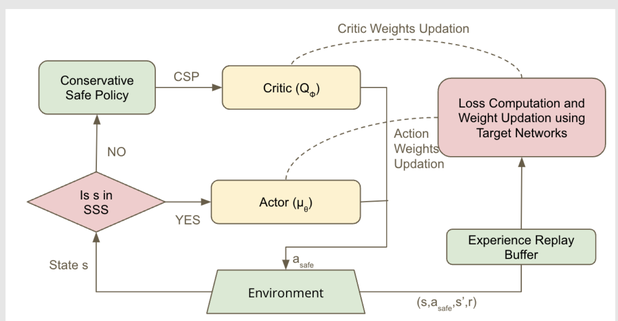


Figure 4. Model Pipeline

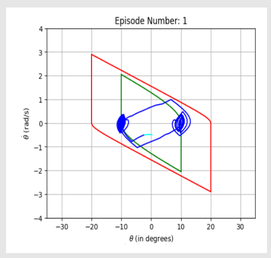
Pseudocode

- Initialise Policy and Critic Parameters (θ and ϕ) of the DDPG Model
- Repeat
  - If $s \in \mathcal{X}_s, a = \pi_\theta(s)$
  - Elif $s \in \mathcal{X}_{ms}, a = argmax_{a' \in \pi_{safe}(s)} Q_\phi(s, a')$
  - The Replay Buffer is populated with the tuples (s, a, s', r)
  - $\nabla_\phi L(\phi) = \nabla_\phi \frac{1}{|B|} \sum \left(r + \gamma Q_{\phi_{\text{targ}}}(s', \mu_{\theta_{\text{targ}}}(s')) - Q_\phi(s, a)\right)^2$
  - $\nabla_\theta L(\theta) = \nabla_\theta \frac{1}{|B|} \sum_{s \in B} (Q_\phi(s, \mu_\theta(s)) + (a - \mu_\theta(s))^2)$
  - The weights are updated using the gradients as defined above
- Until Convergence

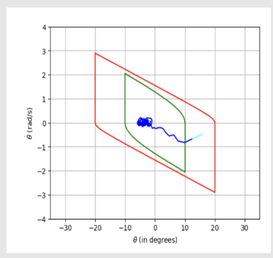
Results

Trajectories during Training and Deployment

- The trajectories of the agent in the environment were tracked during both training and deployment.
- During training, we see that the trajectory often enters the marginally safe region and is immediately pushed back into the safe region using the human provided conservative actions.
- During deployment, we see that the trained agent learns to stay inside the safe region



(a) During Training

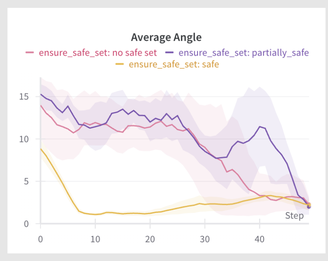


(b) During Deployment

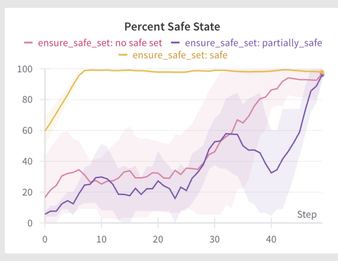
Figure 5. Red: Boundary of $\mathcal{X}_{ms}$ and Green: Boundary of $\mathcal{X}_s$

Safety and Rewards

- The three curves correspond to: No Safe Actions, Partially Safe Actions, Completely Safe Actions
- Our implementation is clearly better than the traditional DDPG algorithm with respect to safety.



(a) Average Angle (Pendulum Environment)



(b) Number of Safe Trajectories as a Percentage

Figure 6. Metrics for Implementations with and without safety layer

Results on Inverted Pendulum

Algorithm	Percent Safe Episodes	Percent Safe Episodes under Disturbances	Average Angle
DDPG Train	74	-	6.19
DDPG Deployment	96	96	4.86
DDPG Safe Layer Train	100	-	4.13
DDPG Safe Layer Deployment	100	96	2.97

Conclusion

- A modified version of the Deep Deterministic Policy Gradient algorithm was implemented with a safety layer for continuous control.
- An additional loss term was included in the policy gradient term to ensure that the agent learns the safety layer.
- The algorithm assumes that the human provides a well-defined SSS and CSP which may not be possible in complex environments with multi-dimensional state space. In this scenario, we are developing a pipeline to estimate SSS and CSP directly from human provided safe trajectories.

References

Ji, J., B. Zhang, J. Zhou, X. Pan, W. Huang, R. Sun, Y. Geng, Y. Zhong, J. Dai, and Y. Yang (2023). "Safety gymnasium: A unified safe reinforcement learning benchmark". In: *Advances in Neural Information Processing Systems* 36.

Brunke, L., M. Greeff, A. W. Hall, Z. Yuan, S. Zhou, J. Panerati, and A. P. Schoellig (2022). "Safe learning in robotics: From learning-based control to safe reinforcement learning". In: *Annual Review of Control, Robotics, and Autonomous Systems* 5, pp. 411–444.

Kazantzidis, I., T. Norman, Y. Du, and C. Freeman (2022). "How to train your agent: active learning from human preferences and justifications in safety-critical environments". In.