Winter Term 21/22

Adversarial Self-Supervised Learning with Digital Twins

Lecture-6: Safe RL

Prof. Dr. Holger Giese (holger.giese@hpi.uni-potsdam.de)

Christian Medeiros Adriano (christian.adriano@hpi.de) - **"Chris"**

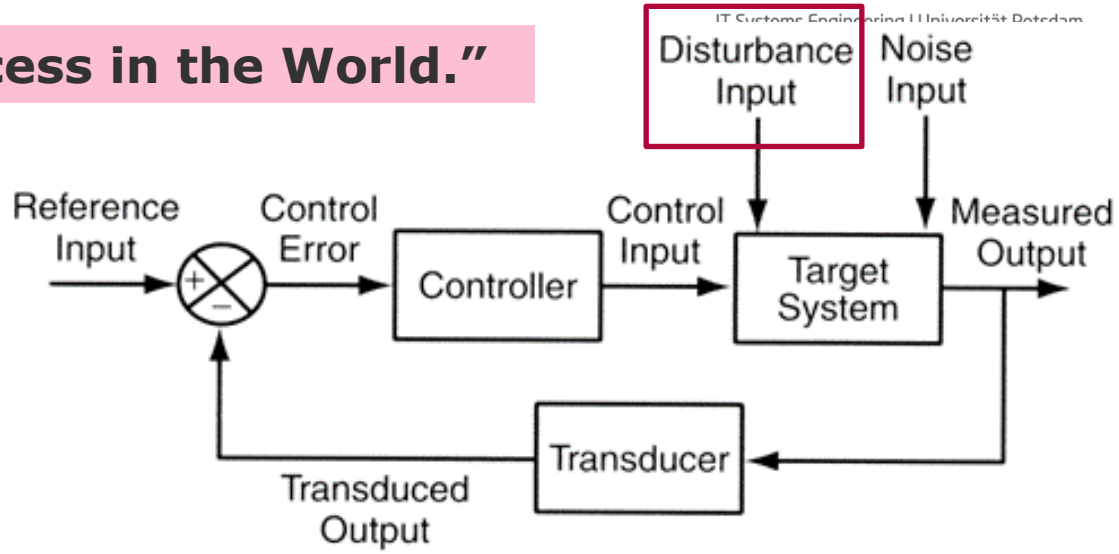
He Xu (he.xu@hpi.de)

Infrastructure to run experiments

“Success in the Lab is not guarantee of Success in the World.”

Feedback loop models

"When solving a problem of interest, do not solve a more general problem as an intermediate step. Try to get the answer that you really need but not a more general one." **Vladimir Vapnik**

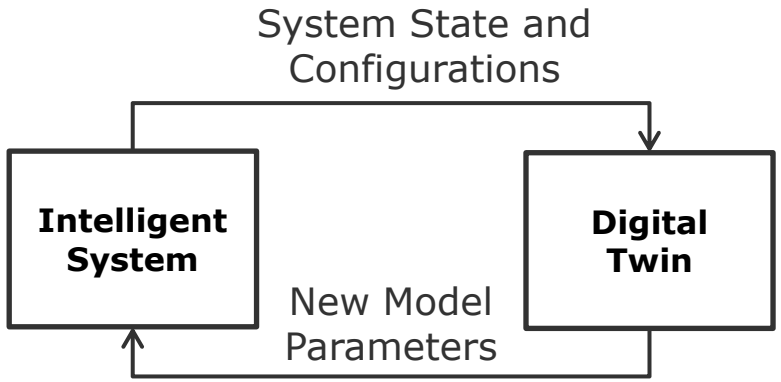


Simulation models

"Thinking is acting in an imagined space" **Konrad Lorenz**

"Perception is a generative act" – [Gross et al. 1999]

" Consciousness is a controlled hallucination " - [Seth et al. 2000]



Success in the World == learn safely or at least be robust to changes

Concrete Problems in AI Safety: Wrong Means to an End

HPI Hasso Plattner Institut
IT Systems Engineering | Universität Potsdam

Avoiding Negative Side Effects: How can we ensure that our cleaning robot will not disturb the environment in negative ways while pursuing its goals. Knocking over a vase because it can clean faster by doing so? **Can we do this without manually specifying everything the robot should not disturb?**

- ***mRubis:*** *While optimizing for risk, make myopic decisions only based on average utility, ignoring the shape of the distribution, which might render some components riskier to fix than others.*

.

source: Amodei et al., 2016, Concrete Problems in AI Safety

Concrete Problems in AI Safety: Wrong Means to an End

Avoiding Reward Hacking: How can we ensure that the cleaning robot won't game its reward function? For example, if we reward the robot for achieving an environment free of messes, it might disable its vision so that it won’t find any messes up or covers over messes with materials it can't see through, or simply hide when humans are around so they can’t see.

- ***mRubis:*** *While comparing the outcomes of multiple agents w.r.t. their cumulative reward, one agent might put the system on a state of constant repair because that would trigger more repair actions, hence more cumulative utility being produced.*

source: Amodei et al., 2016, Concrete Problems in AI Safety

Concrete Problems in AI Safety: Prior Knowledge for Safety?

HPI Hasso Plattner Institut
IT Systems Engineering | Universität Potsdam

Scalable Oversight: How can we efficiently ensure that the cleaning robot respects aspects of the objective that are too expensive to be frequently evaluated during training? For instance, it should throw out things that are unlikely to belong to anyone but put aside things that might belong to someone (it should handle stray candy wrappers differently from stray cellphones). **i.e., can the robot find a way to do the right thing despite limited information?**

- ***mRubis:*** *While optimizing for reward adjusted to risk, how to correctly compute it for rare events? For instance, two real failures masking a third failure, or a cyclic dependency that can only be broken by a complete system restart.*

source: Amodei et al., 2016, Concrete Problems in AI Safety

Concrete Problems in AI Safety – what is safe?

Safe Exploration: How do we ensure that the cleaning robot doesn't make exploratory moves with very bad repercussions? For example, the robot should experiment with mopping strategies, but putting a wet mop in an electrical outlet is a very bad idea.

- ***mRubis:*** *When optimizing for cost, make myopic decisions on number of component instances, which might cause frequent low response time and consequent crashes on depending components.*

source: Amodei et al., 2016, Concrete Problems in AI Safety

Concrete Problems in AI Safety – what is safe?

HPI Hasso Plattner Institut
IT Systems Engineering | Universität Potsdam

Robustness to Distributional Shift: How do we ensure that the cleaning robot recognizes, and behaves robustly, when in an environment different from its training environment? For example, strategies it learned for cleaning an office might be too dangerous on a factory workfloor.

- ***mRubis****: change the distribution of the probabilities in the failure models, so the agent now operates with mismatch model of the environment.*

source: Amodei et al., 2016, Concrete Problems in AI Safety

Challenges of Real-World Reinforcement Learning [Dulac-Arnold 2019]

HPI Hasso Plattner Institut
IT Systems Engineering | Universität Potsdam

1. Training off-line from fixed logs of an external behavior policy
   - pre-train policy & failure model
2. Learning on the real system from limited samples (
   - update policy and failure model)
3. Safety constraints that should never or at least rarely be violated
   - which combination of actions are allowed for a given context (system configuration)
4. Tasks that may be partially observable, alternatively viewed as non-stationary or stochastic.
   - Failure Propagation and Failure Masking POMDP
5. Inference that must happen in real-time at the control frequency of the system.
   - Allow environment and controller to operate in parallel (today they are in lock-step)

Challenges of Real-World Reinforcement Learning [Dulac-Arnold 2019]

Other challenges (not subjected to)

6. High-dimensional continuous state and action spaces.
7. Reward functions that are unspecified, multi-objective, or risk-sensitive.
8. System operators who desire explainable policies and actions.
9. Large and/or unknown delays in the system actuators, sensors, or rewards

Main reason-why:
mRubis is not safety-critical and it does not operate within a physical process

Recap - Sparse, Unsafe, Uncertain World

- Operation produces few if any training examples (relevant events are rare)
- Predictive performance is necessary but not sufficient
- Simulation requirements for robustness are functions of the operational context, hence:
  - Requirements are domain-specific, which make them more challenging to automate
  - Their reification happens outside the machine learning algorithms

Requirements Engineering Questions

1. Which simulation requirements do we need?
2. How do we verify that these simulation requirements are being met?
3. How to incorporate these simulation requirements into training?
4. How do we transfer the simulation outcomes to production?

Deep Neural Network Challenges

Challenge	Description [0]
Brittleness (small deviations in the original input distribution cause incorrect predictions)	Fastening stickers on a stop sign can make an AI misread it [1]. Changing a single pixel on an image can make an AI think a horse is a frog [2]. Neural networks can be 99.99 percent confident that multicolor static is a picture of a lion [3]. Medical images can get modified in a way imperceptible to the human eye so medical scans misdiagnose cancer 100 percent of the time systems can break down in ways that remain a mystery to researchers [4]. **Mitigation** – train NN with "adversarial" examples.
Embedded Bias (unfair decisions made on invalid features)	Nationally deployed health care algorithm in the US was racially biased in favor of white males [5]. **Mitigation** – techniques for detecting unfair decision models
Catastrophic Forgetting (unlearns previous knowledge when learning new knowledge)	When trained to identify new deepfakes, the neural network quickly forgot how to detect the old ones. [6] **Mitigation** - mix new and old data in the training.

Deep Neural Network Challenges

HPI Hasso Plattner Institut
IT Systems Engineering | Universität Potsdam

Challenge	Description [0]
Explainability (the way in which the NN produces certain predictions and not others)	Explanations are important to increase trust in the predictions and also to help debug the NN models. Typical questions are what factors most contributed to a person to be granted a loan or what are the basis of cancer diagnosis? **Mitigation** – Explainability techniques [7] and building databases of correct explanations or search for facts that might explain decisions.
Quantifying Uncertainty (NN can be very certain, even when it is wrong)	In the first Tesla fatal accident [8], the NN was completely certain to be seeing the sky and not a white truck crossing in front of the car [9]. **Mitigation** – Uncertainty Quantification Techniques [10][11], however techniques are too slow, we need faster techniques to be used in real-time safety-critical systems
Common Sense (the ability to reach acceptable, logical conclusions based the everyday knowledge that people take for granted)	NN can learn shortcuts that lead them to misbehave, like misclassifying hate speech, when a human could clearly identify it [12][13]. **Mitigation** – checklists and diverse training data [14]
Math (solve mathematical problems)	A NN was trained on hundreds of thousands of math problems with step-by-step solutions, but when tested on 12,500 problems from high school math competitions, the NN produced only 5% accuracy [15] . **Mitigation** - ?

Challenges of modeling as RL problem

Problem of finding a policy that maximizes cumulative rewards

Need to represent the problem as an $MDP\ (S, A, T, R, \gamma, H)$

Goal: $\max_{\pi} \mathrm{E}[\sum_{t=0}^{H} \gamma^t R(S_t, A_t, S_{t+1}) | \pi]$

Set of states S

Set of actions A

Transition function $P(s' \mid s, a)$

Reward function $R(s, a, s')$

Start state s_0

Discount factor γ

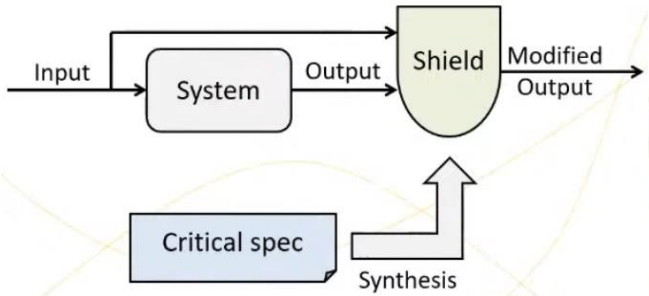
Horizon H

Design decision:

- What are the actions in the MDP?
- How to build a transition matrix?
- How is the reward calculated?
- What is the magnitude of discount horizon H? How many steps ahead should I care to make a decision now?

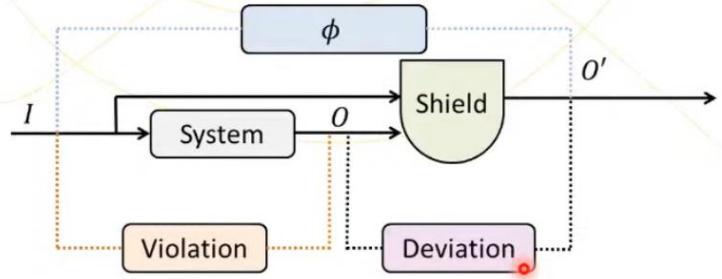
Shielding – Domain-Knowledge for Safe Learning

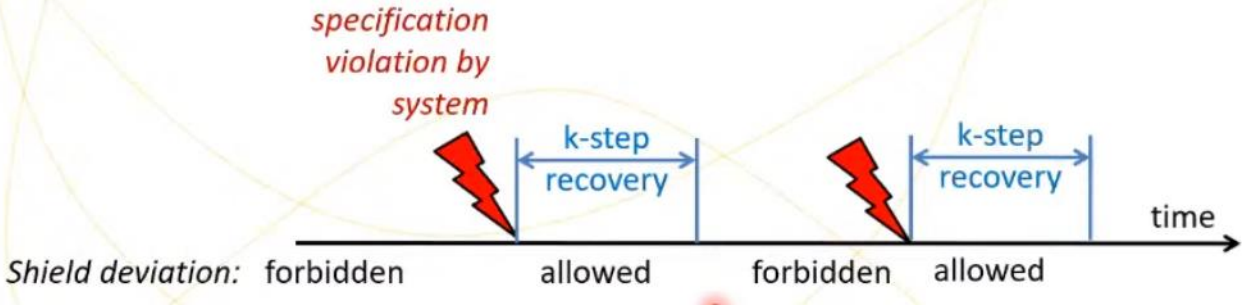
HPI Hasso Plattner Institut
IT Systems Engineering | Universität Potsdam

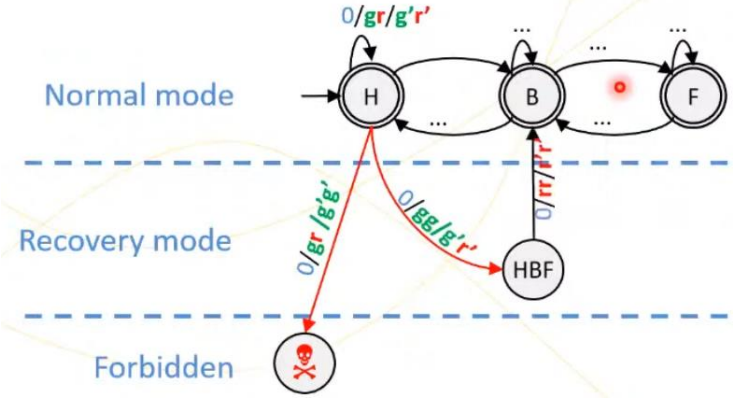


Minimum Interference:

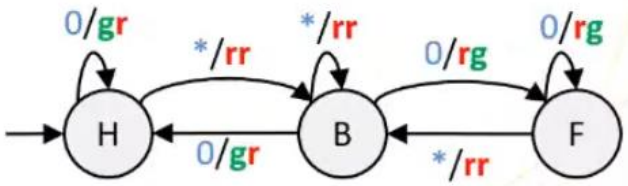
1. Deviate only if necessary
   Retain non-critical properties if system OK
2. Deviate as little as possible
   likely retain non-critical properties if system fails



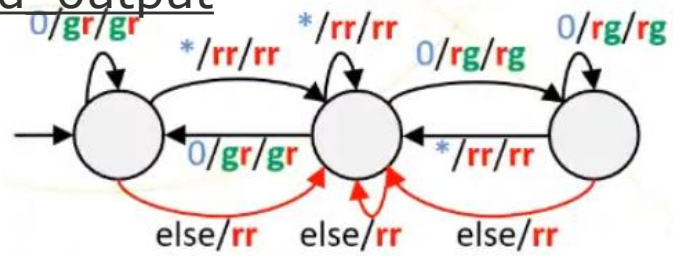




Specification



Label: input / system_output / shield_output



Other extensions:

- Probabilistic shields
- Performance shields
- Timed shields (timed automata)
- Multi-agent shields (global safety)

source: **Shield Synthesis for Safe Reinforcement Learning,** https://www.youtube.com/watch?v=DUkqNJ-rwck

Constrained MDP approach to Safe RL [Tessler 2018]

HPI Hasso Plattner Institut
IT Systems Engineering | Universität Potsdam

Constrained Markov Decision Process (CMDP) extends the MDP framework by introducing a penalty c(s; a), a constraint $C(s_t) = F(c(s_t, a_t), ..., c(s_N, a_N))$

The expectation of the constraint: $J_C^\pi = \mathbb{E}_{s\sim\mu}^\pi[C(s)]$ The objective becomes: $\max_{\pi\in\Pi} J_R^\pi$, s.t. $J_C^\pi \leq \alpha$

Parametrized policies (neural networks), where the parameters of the policy are denoted by θ and a parametrized policy as π_θ. And follow the two assumptions to ensure convergence to a constraint satisfying policy.

Assumption 1. *The value $V_R^\pi(s)$ is bounded for all policies $\pi \in \Pi$.*

Assumption 2. *Every local minima of $J_C^{\pi_\theta}$ is a feasible solution.*

Algorithm 1 Template for an RCPO implementation

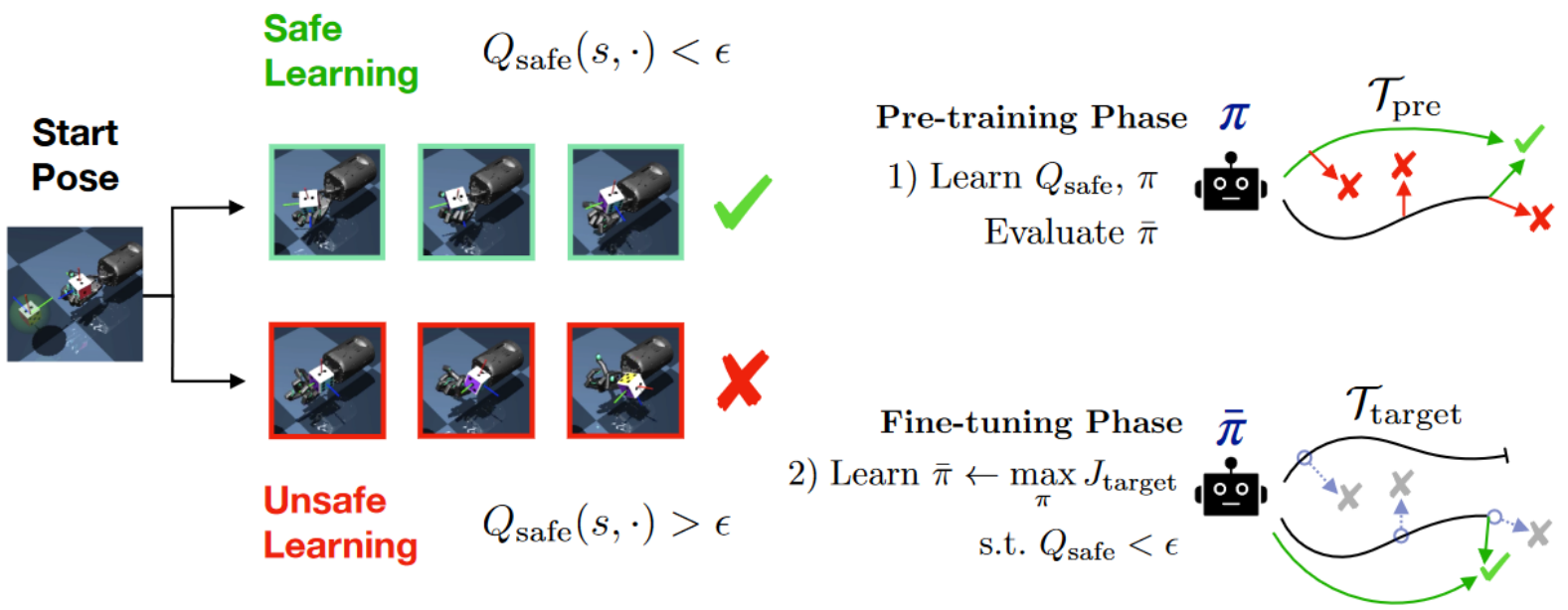
Input: penalty $c(\cdot)$, constraint $C(\cdot)$, threshold α, learning rates $\eta_1(k) < \eta_2(k) < \eta_3(k)$
Initialize actor parameters $\theta = \theta_0$, critic parameters $v = v_0$, Lagrange multipliers and $\lambda = 0$
for $k = 0, 1, ...$ **do**
  Initialize state $s_0 \sim \mu$
  for $t = 0, 1, ..., T-1$ **do**
    Sample action $a_t \sim \pi$, observe next state s_{t+1}, reward r_t and penalties c_t
    $\hat{R}_t = r_t - \lambda_k c_t + \gamma \hat{V}(\lambda, s_t; v_k)$ ▷ Equation 10
    Critic update: $v_{k+1} \leftarrow v_k - \eta_3(k)\left[\partial(\hat{R}_t - \hat{V}(\lambda, s_t; v_k))^2/\partial v_k\right]$ ▷ Equation 11
    Actor update: $\theta_{k+1} \leftarrow \Gamma_\theta\left[\theta_k + \eta_2(k)\nabla_\theta \hat{V}(\lambda, s)\right]$ ▷ Equation 6
  Lagrange multiplier update: $\lambda_{k+1} \leftarrow \Gamma_\lambda\left[\lambda_k + \eta_1(k)\left(J_C^{\pi_\theta} - \alpha\right)\right]$ ▷ Equation 8
return policy parameters θ

Table 1: Comparison between various approaches.

	Handles discounted sum constraints	Handles mean value[1] constraints	Requires no prior knowledge	Reward agnostic
RCPO (this paper)	✓	✓[2]	✓	✓
Dalal et al. (2018)	✗	✗	✗	✓
Achiam et al. (2017)	✓	✗	✓	✓
Reward shaping	✓	✓	✓	✗
Schulman et al. (2017)[3]	✗	✗	✗	✗

Chen Tessler, Daniel J. Mankowitz, and Shie Mannor, 2018, Reward Constrained Policy Optimization

Learning to be Safe with a Safety Critic [Srinivasan 2020]



Q_{safe} : safety critic π : unconstrained policy $\bar{\pi}$: Q_{safe}-constrained policy ✘ : unsafe state ✓ : success state

Pre-training a critic Q^{π}_{safe} that is trained by constraining the actions of a policy π. This yields safe policy iterates (resulting from optimizing Eq. 4) that achieve safe episodes throughout fine-tuning.

Conversely, standard RL approaches will visit unsafe states during adaptation, which results in more failed episodes.

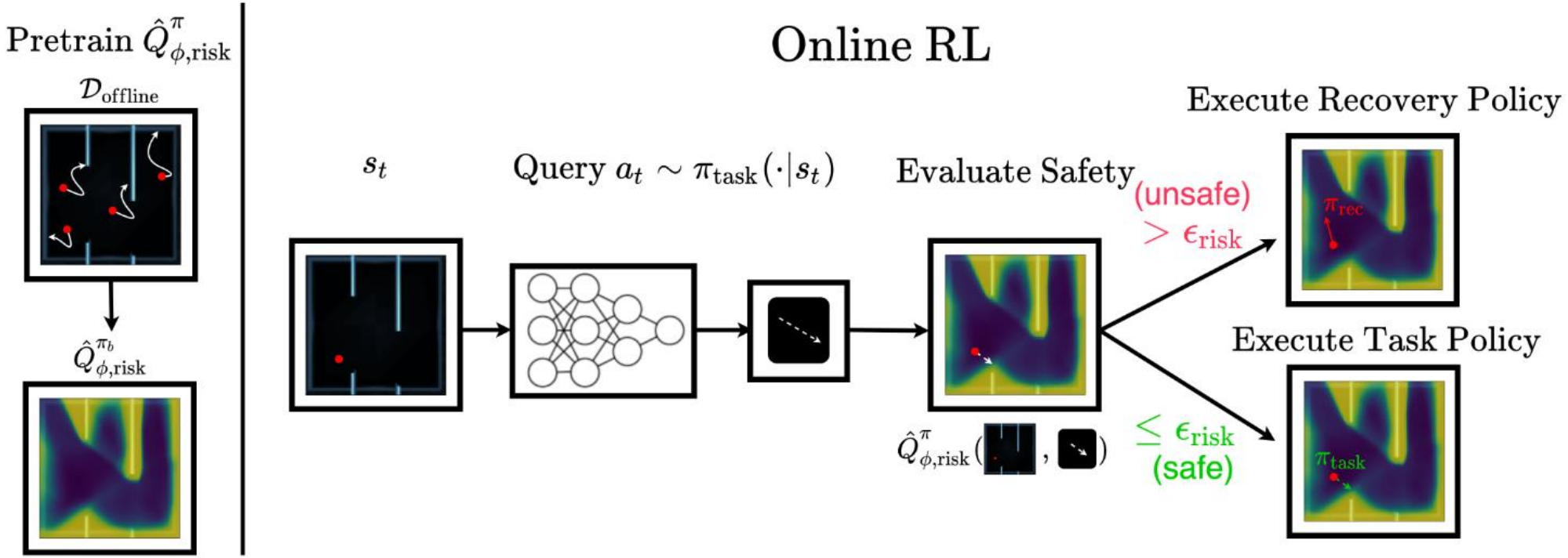
Algorithm 1 SQRL Pre-training

procedure PRETRAIN($n_{\text{pre}}, \mathcal{T}_{\text{pre}}, \epsilon_{\text{safe}}, \gamma_{\text{safe}}$)
  Initialize replay buffers, $\mathcal{D}_{\text{safe}}, \mathcal{D}_{\text{offline}}$
  Initialize networks, $\hat{Q}^{\psi}_{\text{safe}}, \pi_\theta, Q_{\phi_1}, Q_{\phi_2}$.
  $s \sim \mu_{\text{pre}}(\cdot)$
  for n_{pre} steps **do**
    for n_{off} steps **do**
      Sample action, $a \sim \pi_\theta(\cdot|s)$
      $s' \sim P_{\text{pre}}(\cdot|s,a)$
      $\mathcal{D}_{\text{offline}}$.add$((s, s', a, r_{\text{pre}}(s,a)))$
      Apply SAC update to $\theta, \phi_1, \phi_2, \alpha$
      if $\mathcal{I}(s') = 1$ **then** $s' \sim \mu_{\text{pre}}(\cdot)$
      $s \leftarrow s'$
    // Collect k on-policy rollouts
    for $i \leftarrow 1, k$ episodes **do**
      $\tau_i \leftarrow$ ROLLOUT$(\bar{\pi}_\theta, \mathcal{T}_{\text{pre}})$
      $\mathcal{D}_{\text{safe}}$.add$(\tau_i)$
    $\psi \leftarrow \psi - \hat{\nabla}_\psi J_{\text{safe}}(\psi)$
  return $\pi_\theta, \hat{Q}^{\psi}_{\text{safe}}$

Algorithm 2 SQRL Fine-tuning

procedure FINETUNE($n_{\text{target}}, \mathcal{T}_{\text{target}}, \hat{Q}_{\text{safe}}$)
  $s \sim \mu_{\text{target}}(\cdot)$, $\mathcal{D}_{\text{offline}} \leftarrow \{\}$
  for n_{target} steps **do**
    Sample action, $a \sim \bar{\pi}_\theta(\cdot|s)$
    // where $\bar{\pi}_\theta \leftarrow \Gamma(\pi_\theta)$ (Eq. 3)
    $s' \sim P_{\text{target}}(\cdot|s,a)$
    $\mathcal{D}_{\text{offline}}$.add$((s, s', a, r_{\text{target}}(s,a)))$
    // Dual gradient ascent on Eq. 4
    $\theta \leftarrow \theta + \lambda \hat{\nabla}_\theta J_{\text{target}}(\theta, \alpha, \nu)$
    $\alpha \leftarrow \alpha - \lambda \hat{\nabla}_\alpha J_{\text{target}}(\theta, \alpha, \nu)$
    $\nu \leftarrow \nu - \lambda \hat{\nabla}_\nu J_{\text{target}}(\theta, \alpha, \nu)$
    if $\mathcal{I}(s') = 1$ **then** $s' \sim \mu_{\text{target}}(\cdot)$
    $s \leftarrow s'$
  return π_θ

Recovery RL: Safe Reinforcement Learning with Learned Recovery Zones [Thananjeyan 2021]



1- Learn the safety critic $\hat{Q}^{\pi}{}_{\varphi}$,risk with offline data from some behavioral policy π_b, which provides a small number of controlled demonstrations of **constraint violating behavior** as shown on the left.

2- At each timestep, queries the task policy π_{task} for some action *a* at state *s*, evaluates $\hat{Q}^{\pi}{}_{\varphi,\,risk(s,a)}$ and executes the recovery policy π_{rec} if $\hat{Q}^{\pi}{}_{\varphi,\,risk(s,a)} > \varepsilon_{risk}$ and π_{task} otherwise.

Next Tasks (papers are in the Zotero repo)

HPI Hasso Plattner Institut
IT Systems Engineering | Universität Potsdam

1. Safety Challenges in AI and RL
- Dulac-Arnold, Mankowitz, Hester, 2019, Challenges of Real-World Reinforcement Learning
- Amodei et al., 2016, Concrete Problems in AI Safety - https://arxiv.org/abs/1606.06565

2. Choose one of the Safety-RL papers to read and write a gist (8 lines)

- Context (domain and how it might relate to mRubis)
- Problem (what, why it matters, why is it challenging)
- Approach (why is it a good idea/insight, how it works, how it compares with others)

3. Choose one of the Sim2Real papers and write a gist (8 lines)

For tasks 2 and 3, please check the following folders in Zotero:

- Safety (reinforcement learning)
- Sim2Real

END

Further readings recommended

Traoré, R., Caselles-Dupré, H., Lesort, T., Sun, T., Díaz-Rodríguez, N., & Filliat, D. (2019). **Continual reinforcement learning deployed in real-life using policy distillation and sim2real transfer**

How to prevent catastrophic forgetting in sim2real

Xiao, C., Lu, P., & He, Q. (2021**). Flying Through a Narrow Gap Using End-to-End Deep Reinforcement Learning Augmented With Curriculum Learning and Sim2Real**.

IEEE Transactions on Neural Networks and Learning Systems.

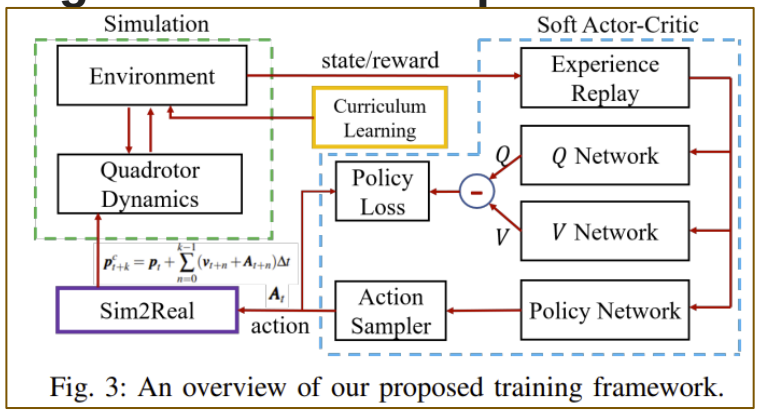


Fig. 3: An overview of our proposed training framework.

Read in more detail and check citations

Kaspar, M., Osorio, J. D. M., & Bock, J. (2020). **Sim2real transfer for reinforcement learning without dynamics randomization.** In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 4383-4388). IEEE.

Kadian, A., Truong, J., Gokaslan, A., Clegg, A., Wijmans, E., Lee, S., ... & Batra, D. (2020). **Sim2Real predictivity: Does evaluation in simulation predict real-world performance?**. *IEEE Robotics and Automation Letters*, *5*(4), 6670-6677.

Balaji, B., Mallya, S., Genc, S., Gupta, S., Dirac, L., Khare, V., ... & Karuppasamy, D. (2019**). Deepracer: Educational autonomous racing platform for experimentation with sim2real reinforcement learning**. *arXiv preprint arXiv:1911.01562*.

Preliminary Work-Packages

1- Hidden Markov Model

Study how to move the HMM implemented in the Python side to the Java side

2- Failure Inject Mechanism

Study how to generate failure injects that reflect that failure propagation patterns

3- Reinforcement Learning

Study how to replace the Supervised Learning controller (Regression) with a Self-Supervised One (RL)

4- Monitoring

Study how to visualize the utility, risk, and mechanism gap between the Real and Simulated (Digital-Twin)

5- Adversarial Tests

Study how to generate stress tests that show how policies that have equivalent predictive outcome at training present distinct outcomes at adversarial test sets

Project Goals - Research Problems

1- Under-specification Problem **Simulation**

Goal: Show that different prediction models solve the task well for testing data, however, perform very differently in two distinct situations:

1.1 distinct hyper-parameters (prior knowledge)

1.2 out-of-distribution data (distribution shifts)

2- Value-at-Risk Problem **Sim2Real**

Goal: Show different rates of synchronization between Production and Simulation can lead to:

2.1 excessive cost of training and redeployment

2.2 increase in the risk of under-performance

3- Learning to Synchronize Problem **Feedback Loop**

Goal: Show that different strategies to learn when to train and redeploy require:

3.1 more data to achieve an average value-at-risk

3.2 longer time to converge