Reward-guided Curriculum for Robust Reinforcement Learning

Siddharth Mysore
Department of Computer Science
Boston University
Boston, MA 02215
sidmys@bu.edu

Robert Platt
Khoury College of Computer Science
Northeastern University
Boston, MA 02115

Kate Saenko
Department of Computer Science
Boston University
Boston, MA 02215

Abstract

We propose a novel method to develop robust action policies using an automated curriculum which seeks to improve task generalization and reduce policy brittleness. Our Reward-guided Curriculum (RgC) self-reflectively chooses what to train on in order to maximize rewards over a task domain, and is a single-policy meta-learning approach designed to augment the training of existing architectures. Experiments on multiple retro Sonic the Hedgehog video games and classical controls tasks indicate notable improvements in task generalization and robustness of the policies trained with RgC. RgC yields a more than 15% improvement over existing baselines on held-out levels in the video game tasks and boosts robustness to changes in dynamics on the controls tasks by more than 10%.

1 Introduction

Deep Reinforcement Learning (RL) techniques have demonstrated laudable performance across a wide array of tasks in games, controls, and more. However, the brittleness and capacity for generalization of trained RL policies remain open problems [16, 12, 6, 18, 29, 30]. These problems are important to address, especially in practical applications, where one might expect varying degrees of distribution shift at test time and where RL agents may encounter states significantly dissimilar from those experienced during training. For example, consider the Sonic the Hedgehog games for the SEGA Genesis (see Figure 1). Each game is made up of a number of different zones and zones are broken up into acts. Different zones present with significant aesthetic difference and every act presents with a different level layout. Such variations in the task domain can severely impair the efficacy of RL agents on unseen environments unless steps are taken to promote generalization over the task distribution.

While a number of previous works attempt to address the sensitivity and generalizability of RL policies [7, 19, 9, 26, 22, 28, 27, 23, 21], we note that they typically sample from a uniform or normal distribution over tasks. Some, such as Ensemble Policy Optimization (EPOpt) [21], go further with post-processing steps to heuristically determine which experiences to keep from a series of rollouts. However, such an approach simply compensates for the sampling strategy being used after the fact,

Preprint. Under review.

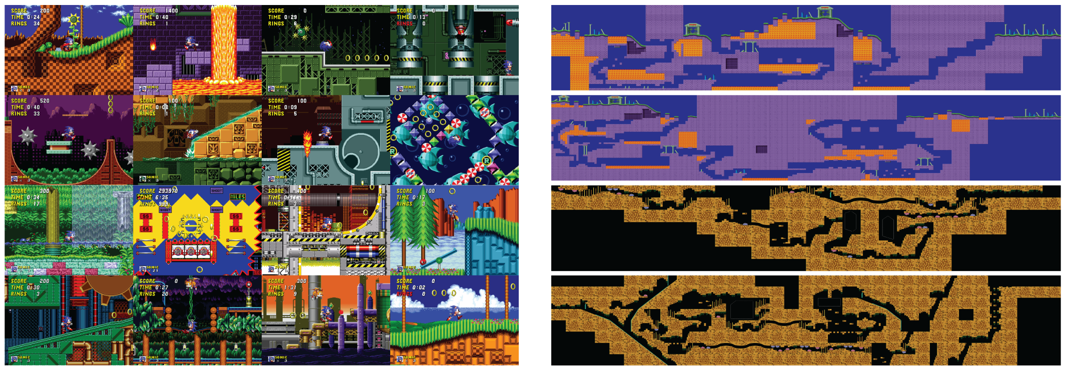


Figure 1: **(Left)** Sampling of frames from each zone in the Sonic the Hedgehog games - Sonic 1 and 2 are considered here (top and bottom 2 rows respectively). Note the visual dissimilarity between different zones, even between zones of the same game. **(Right)** Sampling of different act (level) layouts from the Marble (top) and Labyrinth (bottom) zones. Note that while the levels are visually similar, their layout is significantly different.

as opposed to addressing the sampling strategy itself. In this work we ask: is naive sampling over task variation the best we can do?

In an effort to more intelligently utilize experience gained on a training set and improve robustness and generalization, we develop an approach called *Reward-guided Stochastic Curriculum* (RgC). RgC automatically constructs a training curriculum for each learner to best motivate the learning of robust action policies. We accomplish this by formulating the training curriculum as a multi-armed bandit problem seeking to maximize episodic rewards over the training set, with the bandits guiding sampling probabilities over task distributions. This allows for active control over the tasks sampled during the course of training. RgC is not a RL algorithm in itself, but is rather a meta-learning scheme that is built to automatically guide how RL agents explore the training domain to improve generalization over the task distribution. It is built on the hypothesis that following a training curriculum which allows the agent to acquire more 'useful' experience would result in more robust performance.

The primary advantage of RgC is that by contextualizing past experiences and periodically updating the sampling strategy over the training distribution, RgC constantly pushes the RL agent to explore the task-domain in a way that maximizes expected value gain. This contrasts other naive sampling strategies in that the choice of what experience is gained is an active one and completely under the learner's control. Furthermore, constructing the curriculum as a bandit problem allows learners to adapt sampling strategies during training in a principled way to suit the needs of each agent.

Experiments over multiple tasks show that our method helps agents learn more robust policies, which generalize better and are less brittle to distributional shifts in the task domain when compared to approaches that use simpler sampling strategies [16, 21].

2 Reward-guided Stochastic Curriculum

In this paper, we mainly tackle tasks that present with multiple settings (e.g., game levels, pendulum mass, etc.) and the objective is to develop policies that generalize over all settings. We also want agents to be robust enough to operate under novel tasks settings not experienced during training. We focus specifically on developing model-free RL agents as they do not require any additional knowledge about the task during deployment. Modeling the domain requires prior knowledge of the extent of the task domain, which is not always available at training time.

Our hypothesis is that not all experiences carry equal significance in training. When a person attempts to learn a new skill, they may not remember every aspect of the learning process, but rather focus on the moments that offered the critical insight allowing them to advance their understanding of the skill. We attempt to extend a similar intuition to learning in RL, so that the learning algorithms can automatically identify and prioritize experiences which offer the most value gain. We propose RgC as a way to holistically consider both the past and future impact of choosing to train on any particular instantiation of a task in order to actively inform how tasks are sampled from the training distribution.

RgC draws inspiration from Graves et al. [8], who tackle the problem of multi-task Natural Language Processing with automated curricula. Like them, we formulate the problem of developing an automated curriculum as a multi-armed bandit problem, however we apply it towards RL tasks. We define the curriculum as a sampling policy over the task distribution, where the bandit's choices correspond to determining how tasks should be sampled from the task distribution. To compensate for the needs of each agent being trained, the curriculum also needs to be adaptive. The goal of the bandit is to maximize the payoff of every choice it makes. This would be trivial if the values of each arm are known; however, when choice-values are not known, it is necessary to estimate the value by exploring the task domain and this estimation needs to evolve as new information is gained.

A basic curriculum over N possible task settings can be constructed as an N-armed bandit, with the syllabus of the developed curriculum intended to maximize the reward that the RL agent achieves over the entire task distribution. Over T rounds of 'play', the bandit/curriculum makes a choice, $a_t \in \{1, \dots, N\}$, corresponding to a decision to train under a specific task setting, and observes a payoff r_t, computed as the difference in mean rewards observed before and after training on the selected task setting. The goal of the curriculum is to consistently sample rollouts from the task distribution to maximize learning gains.

To choose settings on which to train in order to minimize regret and maximize overall reward, we employ the Exponentially-weighted algorithm for Exploration and Exploitation (Exp3) [3]. Specifically, we use the Exp3.S variant of the algorithm to develop our multi-armed bandit's policy, which employs an ϵ-greedy strategy and additively mixes weights to ensure that the probabilities of selecting any particular action are not driven to insignificance. We define ϵ to limit the maximum probability of any setting being selected. (*Note*: we present Exp3.S similarly to Graves et al. [8], which is mathematically equivalent to the algorithm as it is presented by Auer et al. [3]).

We further extend the method to attach a different bandit to each different task variable in the training distribution (for example, if mass and volume were variables in a controls task, we would use two bandits - one for mass and one for volume, and each bandit would select different masses or volumes to train on respectively). This multi-multi-armed bandit structure allows for a linear growth of the number of arms to be maintained with the number of variants per variable, as opposed to polynomial with the number of combinations.

A bandit $m \in \boldsymbol{M}$ (where $\boldsymbol{M}$ is a set of task variables) has a policy defined by weights, $w_{m,i}$ $\forall i \in \{1, \dots, N_m\}$, corresponding to the N_m possible task-variable settings. At bandit-step t the bandit chooses setting $a_{m,t} \sim \pi^{\text{Exp3.S}}_{m,t}$, where $\pi^{\text{Exp3.S}}_{m,t}(i_m)$ is the sampling probability of setting i_m:

$$\pi^{\text{Exp3.S}}_{m,t}(i) := (1-\epsilon)\frac{e^{w_{m,i,t}}}{\sum_j e^{w_{m,j,t}}} + \frac{\epsilon}{N_m} \tag{1}$$

At the end of each bandit step, the weights are updated based on observed payoff, r_t:

$$w_{m,t+1,i} := \log\left[(1-\alpha_t)\eta_i + \frac{\alpha_t}{N_m - 1}\sum_{j \neq i}\eta_j\right] \tag{2}$$
$$\text{where } \eta_k = e^{\left(w_{m,t,k} + \hat{r}^{\beta}_{M,t-1,kR}\right)}$$

where $w_{m,1} = 0$, $\alpha_t := t^{-1}$, and the importance sampled payoff is computed as:

$$\hat{r}^{\beta}_{M,t,i} := \frac{r_t \prod_{m \in M} \mathbb{I}_{[a_{m,t}=i_m]} + \beta}{\prod_{m \in M} \pi^{\text{Exp3.S}}_{m,t}(i_m)} \tag{3}$$

To bound the magnitude by which an arm's weight might change at any given step, payoffs, r_t, per bandit step, t are scaled such that $r_t \in [-1, 1]$:

$$r_t := \begin{cases} -1 & \delta R_t < \mu^{20}_t \\ 1 & \delta R_t > \mu^{80}_t \\ \frac{2(\delta R_t - \mu^{20}_t)}{\mu^{80}_t - \mu^{20}_t} - 1 & \text{otherwise} \end{cases} \tag{4}$$

where $\delta R_t = R_t - R_{t-1}$ is the true bandit policy payoff at step t, computed based on mean rewards achieved by the agent on the set of environment setting of interest, and μ^x represents the x^{th} percentile of payoffs achieved: $\{r_{s \le t}\}$. Algorithm 1 outlines the workings of the RgC pipeline.

A key difference in our method from that employed by Graves et al. [8] is in how we define the payoff, i.e. the value gained by training on a specific task setting. Graves et al. perform a comparison

Algorithm 1 Reward-guided Curriculum

```
Initialize: $w_{m,i} = 0 \ \forall\, i \in N_m \ \ \forall\, m \in \boldsymbol{M}$
for $t = 1 \ldots T$ do
    Sample $|\boldsymbol{M}|$ task-variable values $a_{m,t} \sim \pi^{\text{Exp3.S}}_{m,t} \ \forall m \in \boldsymbol{M}$
    Sample $K$ task initializations uniformly from a valid space of initializations
    for $k \in K$ do
        Compute Initial Reward of actor-network policy $p_\theta$ on initialization $k$: $R^{pre}_k$
    end for
    Train network $p_\theta$ on $k \in K$
    for $k \in K$ do
        Compute Post-training Reward of network $p_\theta$ on initialization $k$: $R^{post}_k$
    end for
    Compute learning progress $\delta R_t := mean(\{R^{post}_k - R^{pre}_k\} \ \forall\ k \in K)$
    Map $\delta R_t$ to $[-1, 1]$ by (4)
    Update weights $w_{m,i}$ by (2)
end for
```

on the training loss before and after training, utilizing the same loss metric that is employed by the network. We instead compute our payoff based on the difference in mean episodic rewards over the task distribution before and after training. By doing so, we ensure that the curriculum is based strictly on how the agent is actually performing on the overall set of tasks, as opposed to the agent's own estimation of how well it is performing. Going back to the analogy of a person learning a new skill, RgC's use of the raw reward signals is akin to evaluating performance based on an external exam as opposed to a self assessment. We expect this to be important particularly because we are developing model-free agents, which do not explicitly attempt to model the task environment or its parameters, to generalize to unseen task conditions. In the standard RL setting, where agents are both trained and tested on the same environments, we understand the importance of developing and relying on a good state-value estimation. However, when utilizing agents in task settings that were previously unseen, one should expect that the RL agent's estimations of state-values to be wrong. To mitigate any adverse effects of poor self-evaluation, we choose to have the meta-learner focus on raw reward signals received directly from the tasks in the training distribution.

3 Evaluation

3.1 Baselines

As a point of comparison for RgC, we compare our method against two popular approaches to promote generalization: (i) joint-sampling, and (ii) EPOpt [21].

Joint-sampling offers the simplest basic baseline for a model-free learning approach to handle a distribution of tasks. During training, tasks are sampled uniformly from a training distribution. Randomization techniques, despite their relative simplicity, have been noted to offer notable improvements in the robustness of trained policies [16, 27, 23].

Ensemble Policy Optimization (EPOpt) [21] similarly samples tasks from a task distribution to develop policies in a model-free way. A key difference is that it 'adversarially' prioritizes training on instantiations of tasks with the poorest performances in an effort to seek general improvement. Agents trained with EPOpt-ϵ typically train for a set number of iterations with joint-sampling, after which adversarial experience selection begins. Experiences are gathered in batches of N task rollouts sampled from the task distribution. During the initial phase, all experiences are used for learning. During the adversarial phase, only the worst ϵ fraction of the episodes are used. Our experiments begin the adversarial phase at the halfway point during training, with $\epsilon = 0.2$ and $N = 20$.

3.2 Tasks

To demonstrate the utility of RgC, we investigate the performance impacts of our algorithm on two classes of tasks: (i) playing video games and (ii) classical controls. Video games offer multiple aesthetically and mechanically similar levels, and the level-based structure of games allows us to create distinct training and test sets on which we can test the generalizability of RL agent policies.

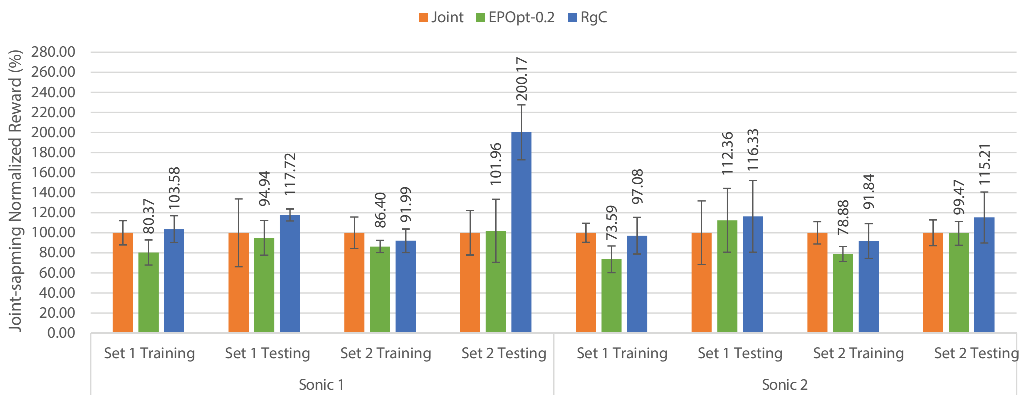


Figure 2: Comparison of rewards achieved over both Sonic games, normalized by the average rewards achieved by the agents trained by Joint-sampling. Rewards achieved were averaged over 15 independently trained agents per set. As a baseline, we present our results relative to Joint-sampling as does nothing more to improve generalizability than simply sampling over the available distribution of training tasks. This allows us to compare the relative value of our method. Training with RgC yields comparable performance to Joint baseline when evaluating agents' performance on their respective training sets. However, when evaluated on a held-out test set, RgC consistently outperforms the baselines, achieving at least a 15% improvement, and even up to 100%.

On the other hand, controls tasks, which model physical interactions, require agents to be robust to variations and inaccuracies in modeling, providing a good test bed for studying policy brittleness.

Sonic Genesis Games The Sonic the Hedgehog games were featured as a part of the OpenAI Retro generalizability contest [16]. As the contest did not make their 3-game training environment publicly available, we consider generalizability across only 2 of the Sonic games individually, with two sets of training and held-out test levels for each game (details on which levels were used for training and testing can be found in the Supplementary Material). We otherwise use the same game interfaces, reward engineering and hyperparameters. We train our agents using the Rainbow [11] network architecture, which is one of the baselines provided by the OpenAI Retro Contest [16]. Here, the task variables that RgC selects from are the levels available in the training set. We evaluate RgC against its joint and EPOpt-0.2 counterparts on *Sonic the Hedgehog* and *Sonic the Hedgehog 2* where two different train/test splits over the game levels are evaluated for each game. Details on how levels are split into training and test sets are provided in the Supplementary Material.

Simple Continuous Controls We also evaluated RgC on a series of classical continuous controls tasks - (i) pendulum balancing, (ii) balancing a cart-pole system, and (iii) 2D-manipulation of a ball on a plane. Tasks (i) and (ii) are derived from OpenAI Gym [5], while (iii) is a custom environment built using the Unity game engine. In all three of these tasks, we sought to investigate the efficacy of agents trained with RgC on tasks involving changing physics and continuous control, where we had previously observed evidence of policy brittleness (details in Supplementary Material). The pendulum and ball-pushing tasks see the mass of the pendulum and ball varied respectively, and the cart-pole has variable cart and pole masses. Our results are compared against: (i) the best results observed via a grid search (oracle) on policies trained exclusively on specific individual task settings (see Supplementary Material), (ii) policies trained with joint-sampling, and (iii) policies trained with EPOpt-0.2. All agents are trained using the Deep Deterministic Policy Gradient (DDPG) [13] architecture, using the hyperparameters proposed in the original paper. When training with RgC, the task variables our curriculum chooses from are the physics settings under which agents are trained. Further details on implementation and task set-up are provided in the Supplementary Material.

3.3 Results

Given the nature of how tasks are sampled during training, the tentative utility and generalizability of trained agents is not clearly reflected in how training rewards evolve over time. RgC chooses tasks based on which ones contribute to overall performance gains and EPOpt chooses tasks which have the worst performances. The effects of these choices on the training rewards signal can be misleading. Instead, we analyze the performance of agents over the entire task distribution - inclusive of both

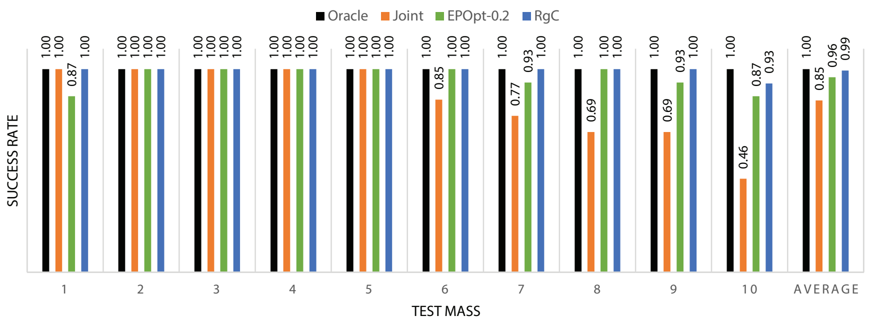


Figure 3: Pendulum policy success rate comparisons [Higher is better]. Success is defined as a deviation of less than 15 degrees from the vertical steady point over the last 100 steps of a 300-step episode. 48 Tests are conducted per agent and the success rate is measured over the performances of 15 agents. Agents are trained only over a distribution of the even-numbered masses. All agents demonstrate an ability to generalize to the lower masses but, with the exception of the oracle (an agent trained only on mass 10), all training methods see a decrease in performance as the test mass increases. However, the RgC success rate does not dip until tested on the highest mass.

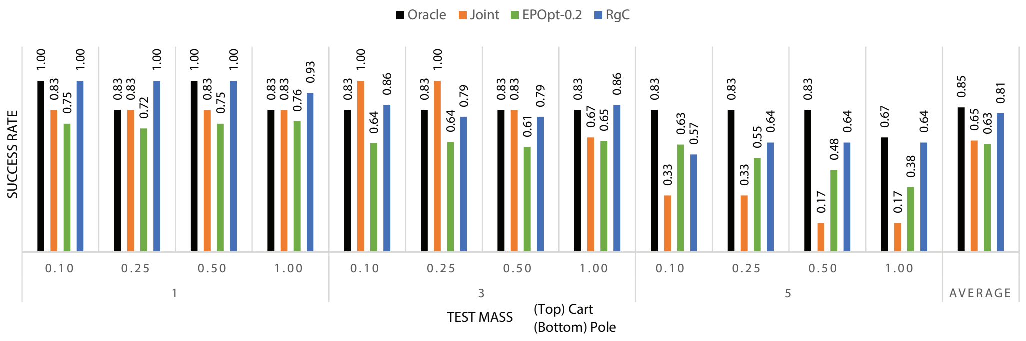


Figure 4: Cart-Pole policy success rate comparisons [Higher is better]. Success is defined as holding the pole steady for at least 490 of 500 steps in an episode. 15 Tests are conducted per agent with random initializations and the success rate is measured over the performances of 15 agents. Agents have access to the full distribution of the task domain during training. The oracle was trained with the highest Cart and Pole masses. It can be noted that all agents' performances drop with an increase in Pole mass but RgC remains relatively robust, maintaining a success rate > 50% over all task settings - which neither of the baselines do.

training and test distributions. Agents trained with RgC generally perform better on average, or at least comparably, when compared to those trained with joint-sampling or EPOpt. Crucially, agents trained with RgC consistently outperform the baselines on the held-out test levels.

In the case of the Sonic games, the disparity in results for the same game under different splits of training and test sets indicates that the choice of levels used in training makes a significant difference to the performance of any agent, as indicated in Figure 2. It should come as no surprise that the quality of the the training data affects the performance of learned policies on test cases. Despite this, RgC oupterforms the baselines on the held-out test sets in all cases.

On the simple controls tasks, our method outperforms policies built on both joint-sampling and EPOpt, and achieves a performance more comparable to our oracle, with the Pendulum and Cart-Pole getting within 1% and 4% of their oracles' success rates respectively (noting that the Pendulum's oracle achieved a 100% success rate). In the case of the ball-pushing task, where we did not have a binary definition of success, it can be noted that the average error is improved over joint-sampling, being within $2\times$ of the oracle's error, as opposed to $3\times$. Our method also has a significantly lower computational cost than the oracle, needing to train only a single policy as opposed to $\prod_{m \in M} N_m$ policies to find the best one.

It is also interesting to note that EPOpt's relative performance over the tasks varies quite significantly, with it outperforming the joint-sampling baseline on the pendulum task and two of the four Sonic game configurations, but performing notably worse on all other tasks. Rajeswaran et al. [21] suggest that a drop in performance is to be expected on the training with EPOpt, but that the algorithm should contribute to lower variance, with lower ϵ values yielding lower variances, while also leading to improved and more stable performance on unseen environments. This claim does not appear to be substantiated by our observations. This inconsistency in the behavior of EPOpt adds further credence to similar observations made by Packer et al. [18], who noted that EPOpt appears to be very sensitive to the tasks at hand and specific network architecture used. It may also further suggest that focusing simply on the experiences where agents perform poorly does not contribute to improvements in general performance. RgC avoids issues such as these by always considering the bigger picture of the entire task distribution (or as much of it as the meta-learner has access to) and choosing tasks which can be expected to improve general performance of the agents. Furthermore, by periodically updating its beliefs on the how experience values are distributed over the task distribution, RgC is able to adapt to the changing needs of the agent as training progresses, as evidenced in Figure 6.

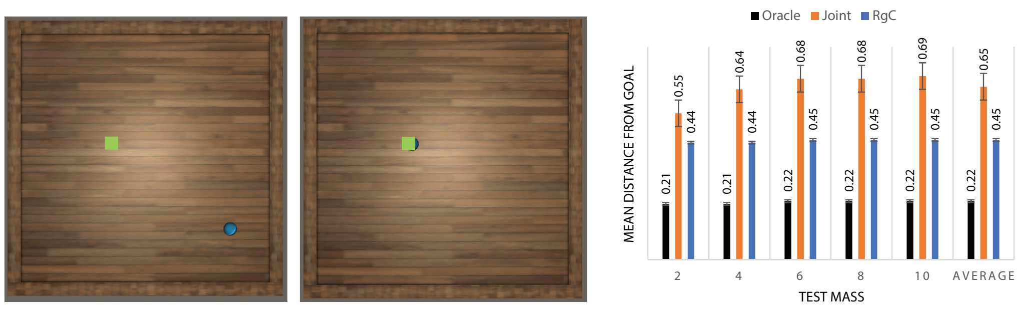


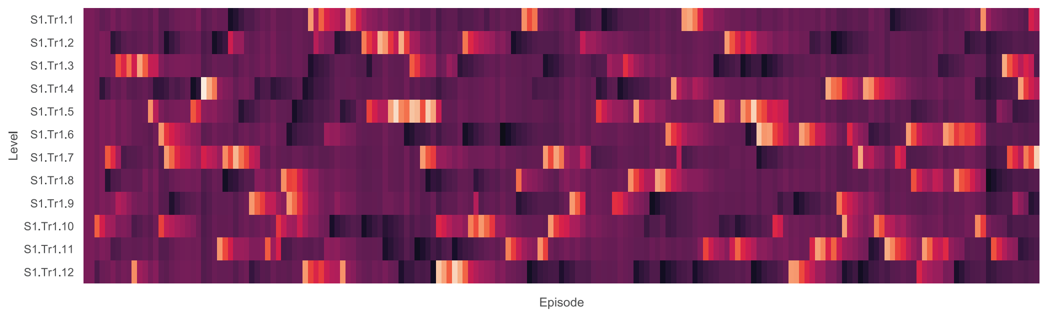
Figure 5: **(Left)** Sample visualization of the ball-manipulation environment. The ball (blue) starts at a randomly initialized point on the plane and the agent's task is to move the ball to the arbitrarily defined goal (green) by imparting the appropriate torques on the ball. **(Middle)** A sample of the ball successfully reaching the goal position **(Right)** Mean error achieved on the ball-pushing control task [Lower is better]. Error bars represent standard deviation of error achieved by 15 separate agents. Agents are trained and tested on the masses listed. The oracle was trained exclusively on the lowest mass. While RgC does not match the performance of the oracale, it still outperforms agents trained by Joint-sampling. *Note: EPOpt comparison not reported as, at the time of writing, we were unable to train enough agents to provide comparable statistical significance.*

4 Related Work

With the advent of deep RL, significant progress has been made over the past decade towards solving complex tasks. With that, the problem of generalization in RL has received increased attention from the broader RL community - with a specific focus on developing agents that are robust to distributional shifts in the task domain [10, 16, 12, 6, 18, 29, 30].

Many of the existing model-free approaches to improving generalization appear to share a core idea: expose the agent to all the relevant task variations, attempt to account for different value associations that different tasks might encourage, and hope that some level of generalization might be achievable. Despite its simplicity, the relative efficacy of this type of straightforward approach has been well documented [16, 27, 18, 23]. The OpenAI Gym Retro Challenge [16] was conducted in an attempt to systematically test algorithms' abilities to generalize to unseen environments by having agents play levels of Sonic the Hedgehog. Interestingly, the best-performing approaches were achieved primarily by tuning the baselines – 'joint PPO' and 'joint Rainbow', which are based on the PPO [25] and Rainbow [11] architectures respectively. This form of 'joint' sampling forms the basis of one of the key baselines utilized in this paper.

A number of meta-learning techniques recognize the inequality in the importance of different experiences during training and attempt to intrinsically weigh or augment experiences gained relative to their expected utility [2, 21, 24, 7, 26]. Some methods adversarially motivate generalization. EPOpt [21], the other baseline which we compare RgC to, attempts to tackle the problem of generalization by explicitly focusing on episodes with relatively poor rewards. In the context of control systems, Mandlekar et al. [14] and Pinto et al. [20] address the issue of policy brittleness in the face of



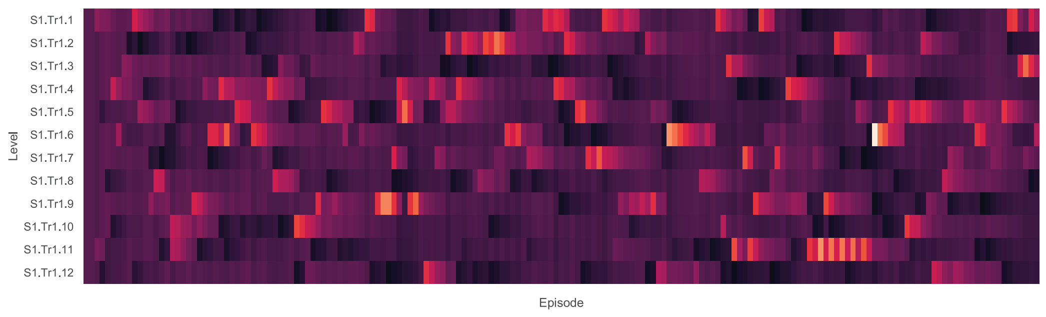


Figure 6: Sample heat-maps to visualize how the sampling probabilities over the training distribution evolves during the course of training two independent agents with RgC for Sonic the Hedgehog 1 - train/test split 1. Observe that despite starting with a uniform sampling distribution, the meta-learner eventually begins to focus on specific levels and constantly adjusts its sampling per the needs of the agent being trained. Furthermore, the sampling strategies are different between the two learners, despite operating on the same task distribution, indicating that the meta-learner adapts the the needs of individual learners

environment perturbation by exposing the policies to plausible physical perturbations during training. While this approach does allow for the development of robust policies, by the authors' own admission however, such approaches may result in poor-performing policies and presents with high variance in performance.

Curriculum learning has been demonstrated as a powerful strategy in improving learning performance over multiple tasks [4, 17, 1], however curricula are often partially or completely hand crafted. An approach that is philosophically similar to ours is that of Matiisen et al. [15] who similarly utilize reward signals to guide task selection. They differ significantly in how the reward information is used however. Where we use reward signals to update bandit policies, Matiisen et al. opt to use linear regression to update the task sampling distribution. Comparing RgC against their technique would make for an interesting comparison in future work.

5 Conclusion

We proposed RgC, a meta-learning technique learning using automatic stochastic curricula, guided by reward signals on a task distribution, to get the most out of an agent's training environment. The curricula developed adapt to the experiences of each learner, allowing for a notion of self-reflection and self-correction. By conducting experiments on a two different video games with two sets of training and test levels for cross validation, We observe a significant improvement in generalizability, with agents achieving more than a 15% improvement over more naive sampling strategies. RgC yields similar improvement in success rates on a series of control tasks, demonstrating reduced policy brittleness when faced with physical variations in a series of classical controls tasks.

We do however recognize some limitations of our work. The current assumption that the environment can be discretized in the manner presented cannot be expected to hold in general. Furthermore, while performance gains under the objective of generalizability are significant, it is important to recognize that RgC incurs additional computational costs in periodically evaluating the value of the sampling strategy being utilized. Though none of the tasks considered in this paper contributed to significant overhead, it is a fact that computational cost scales with the degree of variability in the task settings - with more variation requiring more checks. Should the algorithm be adapted to work with more diverse tasks, it would be important to consider how the meta-learning payoff can be computed efficiently. Future work would investigate techniques which could extend the principles of RgC to continuous variation in task parametrization and improve the efficiency with which meta-learning payoff signals are computed.

References

[1] Jacob Andreas, Dan Klein, and Sergey Levine. Modular multitask reinforcement learning with policy sketches. *CoRR*, abs/1611.01796, 2016.

[2] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5048–5058. Curran Associates, Inc., 2017.

[3] Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. The nonstochastic multiarmed bandit problem. *SIAM J. Comput.*, 32(1):48–77, January 2003.

[4] Yoshua Bengio, Jerome Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In Leon Bottou and Michael Littman, editors, *Proceedings of the Twenty-sixth International Conference on Machine Learning (ICML'09)*. ACM, 2009.

[5] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *CoRR*, abs/1606.01540, 2016.

[6] Karl Cobbe, Oleg Klimov, Christopher Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. *CoRR*, abs/1812.02341, 2018.

[7] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[8] Alex Graves, Marc G. Bellemare, Jacob Menick, Rémi Munos, and Koray Kavukcuoglu. Automated curriculum learning for neural networks. *CoRR*, abs/1704.03003, 2017.

[9] Abhishek Gupta, Benjamin Eysenbach, Chelsea Finn, and Sergey Levine. Unsupervised meta-learning for reinforcement learning. *CoRR*, abs/1806.04640, 2018.

[10] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *AAAI*, 2018.

[11] Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Daniel Horgan, Bilal Piot, Mohammad Gheshlaghi Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. *Association for the Advancement of Artificial Intelligence Conference on Artificial Intelligence*, 2017.

[12] Jan Leike, Miljan Martic, Victoria Krakovna, Pedro A. Ortega, Tom Everitt, Andrew Lefrancq, Laurent Orseau, and Shane Legg. AI safety gridworlds. *CoRR*, abs/1711.09883, 2017.

[13] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *International Conference on Learning Representations*, 2016.

[14] Ajay Mandlekar, Yuke Zhu, Animesh Garg, Fei-Fei Li, and Silvio Savarese. Adversarially robust policy learning: Active construction of physically-plausible perturbations. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 09 2017.

[15] Tambet Matiisen, Avital Oliver, Taco Cohen, and John Schulman. Teacher-student curriculum learning. *CoRR*, abs/1707.00183, 2017.

[16] Alex Nichol, Vicki Pfau, Christopher Hesse, Oleg Klimov, and John Schulman. Gotta learn fast: A new benchmark for generalization in RL. *CoRR*, abs/1804.03720, 2018.

[17] Junhyuk Oh, Satinder P. Singh, Honglak Lee, and Pushmeet Kohli. Zero-shot task generalization with multi-task deep reinforcement learning. In *ICML*, 2017.

[18] Charles Packer, Katelyn Gao, Jernej Kos, Philipp Krähenbühl, Vladlen Koltun, and Dawn Xiaodong Song. Assessing generalization in deep reinforcement learning. *CoRR*, abs/1810.12282, 2018.

[19] Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. DeepMimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Trans. Graph.*, 37(4):143:1–143:14, July 2018.

[20] Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. *CoRR*, abs/1703.02702, 2017.

[21] Aravind Rajeswaran, Sarvjeet Ghotra, Balaraman Ravindran, and Sergey Levine. EPOpt: Learning robust neural network policies using model ensembles. *International Conference on Learning Representations*, 2017.

[22] Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *CoRR*, abs/1606.04671, 2016.

[23] Fereshteh Sadeghi, Alexander Toshev, Eric Jang, and Sergey Levine. Sim2Real view invariant visual servoing by recurrent control. *CoRR*, abs/1712.07642, 2017.

[24] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *International Conference on Learning Representations*, 2016.

[25] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

[26] Bradly C. Stadie, Ge Yang, Rein Houthooft, Peter Chen, Yan Duan, Yuhuai Wu, Pieter Abbeel, and Ilya Sutskever. The importance of sampling in meta-reinforcement learning. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada.*, pages 9300–9310, 2018.

[27] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 23–30, 09 2017.

[28] Wenhao Yu, C. Karen Liu, and Greg Turk. Preparing for the unknown: Learning a universal policy with online system identification. *Robotics: Science and Systems*, 2017.

[29] Amy X. Zhang, Nicolas Ballas, and Joelle Pineau. A dissection of overfitting and generalization in continuous reinforcement learning. *CoRR*, abs/1806.07937, 2018.

[30] Chiyuan Zhang, Oriol Vinyals, Rémi Munos, and Samy Bengio. A study on overfitting in deep reinforcement learning. *CoRR*, abs/1804.06893, 2018.