# Hyperparameters in Contextual RL are Highly Situational

**Theresa Eimer**[*], **Carolin Benjamins**[*] **and Marius Lindauer**
Leibniz University Hanover
{eimer,benjamins}@tnt.uni-hannover.de

## Abstract

Although Reinforcement Learning (RL) has shown impressive results in games and simulation, real-world application of RL suffers from its instability under changing environment conditions and hyperparameters. We give a first impression of the extent of this instability by showing that the hyperparameters found by automatic hyperparameter optimization (HPO) methods are not only dependent on the problem at hand, but even on how well the state describes the environment dynamics. Specifically, we show that agents in contextual RL require different hyperparameters if they are shown how environmental factors change. In addition, finding adequate hyperparameter configurations is not equally easy for both settings, further highlighting the need for research into how hyperparameters influence learning and generalization in RL.

## 1 Introduction

Even though reinforcement learning (RL) has shown considerable progress in many areas like game playing [33, 4], robot manipulation [20], traffic control [2], chemistry [36] and logistics [21], deploying RL in application remains challenging. This is especially in high-stakes domains like autonomous driving and healthcare where failures can be fatal. One explanation is that the design of modern RL agents does not prioritize generalization, making them susceptible to even small variations in their environment or hyperparameters [15, 17, 23, 24].

Different alterations of an environment and the impact on the agents' performance can be modelled and studied e.g. via contextual RL. For instance, in OpenAI's pendulum environment [6] the task is to exert force upon a pendulum such that it balances upright. The other factors defining this process, often physical attributes like the length of the balancing pole, its mass or even the magnitude of gravity, can be chosen as context features. By varying context features during training and testing we are able to represent different instances of the same environment, see Figure 1. This will challenge the agent to perform well across environmental changes and ultimately gives a better estimate of its robustness and generalization capabilities, a step closer to reliable real-world application of RL [5].

While cRL opens many new research directions, it also introduces more variables into an already brittle RL pipeline. From the perspective of a RL practitioner, this can further complicate the process of finding an agent that solves a given task - and indeed we easily show it does. Our research hypotheses are:

1. Hyperparameter configurations for RL agents in the contextual setting are very sensitive to even small changes in the task.
2. Well-performing hyperparameter configurations for RL agents depend on whether context features are explicitly provided or not.

[*]Equal Contribution, Contact Author

We study validity of these research hypotheses in our experiments on several contextually extended environments from the benchmark library CARL [5]. Our results demonstrate the importance of developing and applying AutoRL methods to tune existing RL methods in accordance with their task as well as generating better insights into where and why RL agents fail.

## 2 Related Work

The sensitivity of RL algorithms to changes in their hyperparameters [15, 17] has been a first indicator for the need of hyperparameter optimization. Research into which hyperparameters and algorithm components are deciding factors for different classes of algorithms like policy gradient [10, 1] or off-policy algorithms [26] has contributed to our fundamental understanding of RL algorithms, even though it has not solved their instability issues.

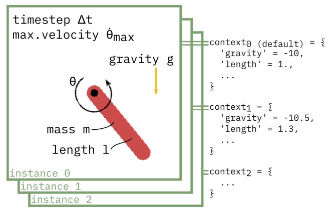
Automatically learning or tuning RL pipelines for a given problem can instead significantly boost performance. There is a broad span of methods, from learning RL algorithms from scratch [34, 7], to learning algorithm components [11, 8] or tuning the agent's hyperparameters [19, 16, 18, 12].

While prior work on hyperparameters in RL shows that RL algorithms are sensitive to hyperparameters and greatly benefit from optimized hyperparameters settings, aptitudes of hyperparameter configurations on changing or varying training environments are underexplored. When the goal is robust RL, however, it is just as important to determine how sensitive the hyperparameter configuration is to environment perturbations as to find a well-performing configuration.

## 3 Contextual Reinforcement Learning

A contextual Markov Decision Process (cMDP) [14, 25] is defined as a set of multiple MDPs $\mathcal{M}_\mathcal{I} := \{\mathcal{M}_i\}_{i \sim \mathcal{I}}$ characterized by the instance $i$ sampled from the instance set or distribution $\mathcal{I}$. The MDP is a 4-tuple $\mathcal{M}_i := (\mathcal{S}, \mathcal{A}, \mathcal{T}_i, \mathcal{R}_i)$ consisting of a state space $\mathcal{S}$, action space $\mathcal{A}$, transition function $\mathcal{T}_i$ and reward function $\mathcal{R}_i$. The transition function $\mathcal{T}_i$ and reward function $\mathcal{R}_i$ are subject to change across instances. With this formulation we can express slight variations in the environment, e.g., varying lengths and masses of a pendulum, and therefore train for generalization. The objective can vary according to the current application, e.g. maximizing the expected return across a test instance set or for a single, hard instance.

In order to analyze the effect of HPs in this setting we use the CARL (Contextually Adaptive RL) benchmark library [5]. CARL extends well-known environments and makes the context defining the behavior of the environment configurable and optionally visible. The context is often based on physical parameters like gravity or friction, see Figure 1. In our experiments, we use these benchmarks to generalize over different instances (contexts) of the same environment, drawn from a common context distribution. The benchmark includes environments from Open AI's gym [6] (classic control and box2d), Google Brax' [13] locomotion environments as well as Super Mario (TOAD-GAN) [3, 30] controlling level similarity and a RNA folding environment [29].



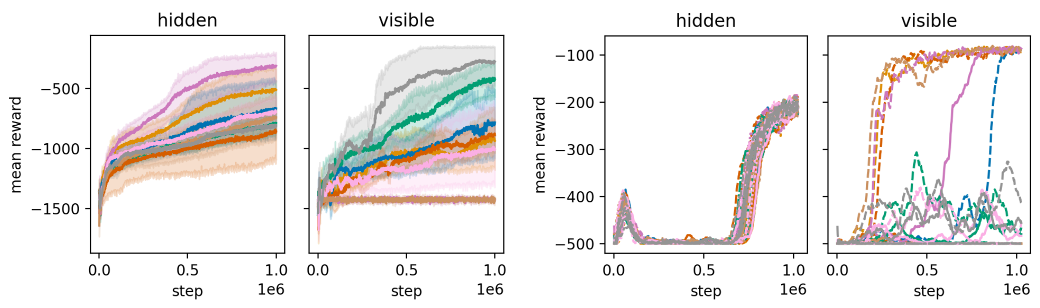
```
context0 (default) = {
  'gravity' = -10,
  'length' = 1.,
  ...
}
context1 = {
  'gravity' = -10.5,
  'length' = 1.3,
  ...
}
context2 = {
  ...
}
```

Figure 1: In the contextual RL setting Pendulum's [6] physical parameters are varied across instances while the underlying dynamic equations stay the same.

## 4 Experiments

We evaluate how context and changes in environments influence the meta-problem of setting the hyperparameters. For this purpose, we use PB2 [18] to optimize the hyperparameters of a standard DDPG [22] algorithm on CARLPendulum and PPO [31] on CARLAcrobot and CARLLunarLander [5]. For all environments we first use 8 parallel PB2 workers on one seed[2] to optimize the hyperparameters during the training with a time budget of $24\,\text{h}$ and $150\,\text{GB}$ of memory. For DDPG,

[2]We note that one PB2 run with a single seed only allows us to draw preliminary conclusions from our results and further runs are needed to verify our findings. Nevertheless, we believe that our results provide an important first indication for potential challenges in AutoRL for cRL.



(a) CARLPendulum

(b) CARLAcrobot

Figure 2: Training performance with hidden and visible context on each hyperparameter schedule found by PB2.The different line colors indicate schedules found by individual PB2 workers, left with confidence intervals, right with one line for each seed.

we optimize the learning rate, discount factor and soft update $\tau$; for PPO the learning rate, discount factor, entropy and value function coefficients, the maximum gradient normalization and GAE parameter. The optimization bounds of the hyperparameters can be found in Appendix A. For testing the hyperparameter policies we use 5 new seeds. Please see the Appendix A for further details on the hardware, software and hyperparameters. The code for these experiments along with the found hyperparameter schedules is available at a anonymous repository during the review period and the actual one will be public upon acceptance: `https://github.com/automl-private/cRL_HPO`

## 4.1 Tunability of Visible and Hidden Context Features

Using the setup from above we varied the gravity for CARLPendulum and the length of the first link for CARLAcrobot. Similar trends can be observed for both environments. First of all, visible context information in combination with hyperparameter optimization leads to better overall performance and even learning speed. For CARLPendulum a final evaluation score of $-312$ was achieved for hidden context and $-262$ for visible context. For CARLAcrobot the difference is more substantial: $-184.65$ for hidden, $-80.13$ for visible context, on the best seed, resp.

Our second observation is that hyperparameter optimization on visible context information seems to be much harder. In particular on CARLAcrobot, only some seeds of the hyperparameter schedule evaluation are able to perform well and most of them fail. There is not only a large performance gap between schedules, but also between different random seeds which we do not observe if we hide the context, see lines with same colors in Figure 2b. On CARLPendulumn, there is also some performance spread for hidden context information, but the spread is wider for visible context information.

Last but not least, we also looked into some of the hyperparameter schedules found by PB2, see Figure 4 in the appendix. These schedules change more for visible than for hidden context information. This is another indication that hyperparameter optimization for cRL with visible context information is harder, although more explicit information is provided.

## 4.2 A Failure Case: LunarLander

With the same experimental setup as before, we optimzed the hyperparameters on the CARLLunarLander environment across varying gravity settings. The results paint a different picture than above (see Figure 3): While PB2 was able to find hyperparameter schedules resulting in a positive performance trend for hidden contexts, it could not do so for visible contexts. In addition, although the agent with the hidden contexts reaches higher rewards, it was still not able to solve the environment.

There are several possible reasons for these observations. We believe that the most likely ones include: (i) The variation introduced into CARLLunarLander via the different contexts might be much larger or much harder to learn than in CARLPendulum and CARLAcrobot. In the case of hidden context information, the agent might not be able to sufficiently distinguish between different instances. The sub-optimal performance even in the case of hidden context information compared to the solved

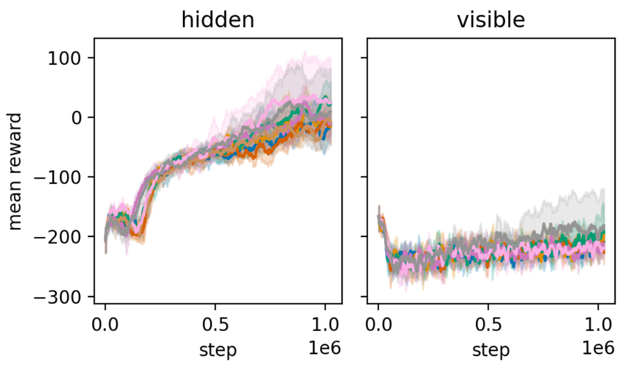


Figure 3: Training performance with hidden and visible context on each hyperparameter schedule found by PB2 on CARLLunarLander. The different line colors indicate schedules found by individual PB2 workers with confidence intervals.

score of 200 is an indication for that. In the case of visible context information, the agent is able to distinguish between the instances, but cannot properly learn to solve the environment potentially due to catastrophic forgetting, which cannot be compensated by HPO alone. (ii) The visible context information provides so much added information to process, that the capacity of the default policy network might be insufficient. Therefore the agent cannot learn when the context is visible, but improves at least to a degree when it is hidden. This would imply that neural architecture search [9] would become much more important for RL. Some further preliminary results supporting this are shown in Appendix C. (iii) For each instance, a different optimal hyperparameter configuration is needed and there is no single hyperparameter configuration that performs well on all the tasks. In the related field of algorithm configuration this is a typical observation for some tasks [35]. (iv) Because HPO is much harder in this case (for unknown reasons), the hyperparameter optimization could not move past a local optimum in the learning rate (see Appendix A) for the visible case, resulting in a suboptimal hyperparameter configuration. This hypothesis would also be supported by the results on CARLPendulum and CARLAcrobot.

At the moment, we cannot say for certain which of these reasons caused our experiments to fail or if more factors are involved. Our findings, however, are a further indication that the cRL setting, especially as environments grow more challenging, is harder to navigate for AutoRL methods compared to standard RL.

## 5 Conclusion and Future Work

For the contextual RL setting we show that tuning hyperparameters for the RL algorithm at hand plays a crucial role in solving the environment. Especially in the case where we provide the agent with the context, the hyperparameter configuration can decide between success and failure. Even so, using hyperparameter optimization in cRL is more complex than for standard RL tasks and as we have seen, finding well-performing configurations is not a guarantee. The addition of context makes the learning process more difficult and harder to optimize for and finding the source of failure can be hindered by the fact that we have comparatively little knowledge about the cRL setting. Thus there is a need for more insights into cRL in general and AutoRL for cRL in particular.

With these preliminary results we want to motivate further research both into how RL agents generalize in the first place and how we can more easily adapt their hyperparameters to support this goal. Our results also clearly show the importance of using existing AutoRL methods when applying RL to a task and the potential improvements gained. Lastly, they emphasise that the methods we currently use for RL are constrained to their problem setting by factors including their hyperparameters. If we want to move beyond single task settings and towards general RL, we need to refine our understanding of the interplay of RL algorithms and their hyperparameters and develop efficient AutoRL practices.

## Acknowledgements

The authors acknowledge funding by the German Research Foundation (DFG) under LI 2801/4-1.

## References

[1] Andrychowicz, M., Raichuk, A., Stanczyk, P., Orsini, M., Girgin, S., Marinier, R., Hussenot, L., Geist, M., Pietquin, O., Michalski, M., Gelly, S., and Bachem, O. (2021). What matters for on-policy deep actor-critic methods? A large-scale study. In *9th International Conference on Learning Representations, ICLR 2021*. OpenReview.net.

[2] Arel, I., Liu, C., Urbanik, T., and Kohls, A. (2010). Reinforcement learning-based multi-agent system for network traffic signal control. *IET Intelligent Transport Systems*, 4(2):128–135.

[3] Awiszus, M., Schubert, F., and Rosenhahn, B. (2020). TOAD-GAN: Coherent style level generation from a single example. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 16.

[4] Badia, A., Piot, B., Kapturowski, S., Sprechmann, P., Vitvitskyi, A., Guo, Z., and Blundell, C. (2020). Agent57: Outperforming the atari human benchmark. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 507–517. PMLR.

[5] Benjamins, C., Eimer, T., Schubert, F., Biedenkapp, A., Rosenhahn, B., Hutter, F., and Lindauer, M. (2021). CARL: A benchmark for contextual and adaptive reinforcement learning. *ArXiv abs/2110.02102*.

[6] Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. (2016). Openai gym. *CoRR*, abs/1606.01540.

[7] Co-Reyes, J. D., Miao, Y., Peng, D., Real, E., Le, Q. V., Levine, S., Lee, H., and Faust, A. (2021). Evolving reinforcement learning algorithms. In *Proceedings of the International Conference on Learning Representations (ICLR'21)*. OpenReview.net. Published online: `iclr.cc`.

[8] Duan, Y., Schulman, J., Chen, X., Bartlett, P., Sutskever, I., and Abbeel, P. (2016). Rl\$ˆ2\$: Fast reinforcement learning via slow reinforcement learning. *CoRR*, abs/1611.02779.

[9] Elsken, T., Metzen, J. H., and Hutter, F. (2019). Neural architecture search: A survey. *Journal of Machine Learning Research*, 20(55):1–21.

[10] Engstrom, L., Ilyas, A., Santurkar, S., Tsipras, D., Janoos, F., Rudolph, L., and Madry, A. (2020). Implementation matters in deep RL: A case study on PPO and TRPO. In *8th International Conference on Learning Representations, ICLR*. OpenReview.net.

[11] Finn, C., Abbeel, P., and Levine, S. (2017). Model-agnostic meta-learning for fast adaptation of deep networks. In Precup, D. and Teh, Y., editors, *Proceedings of the 34th International Conference on Machine Learning (ICML'17)*, volume 70, pages 1126–1135. Proceedings of Machine Learning Research.

[12] Franke, J., Köhler, G., Biedenkapp, A., and Hutter, F. (2021). Sample-efficient automated deep reinforcement learning. In *9th International Conference on Learning Representations, ICLR*. OpenReview.net.

[13] Freeman, C. D., Frey, E., Raichuk, A., Girgin, S., Mordatch, I., and Bachem, O. (2021). Brax - A differentiable physics engine for large scale rigid body simulation. *CoRR*, abs/2106.13281.

[14] Hallak, A., Castro, D. D., and Mannor, S. (2015). Contextual markov decision processes. *arXiv:1502.02259 [stat.ML]*.

[15] Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., and Meger, D. (2018). Deep reinforcement learning that matters. In McIlraith, S. and Weinberger, K., editors, *Proceedings of the Conference on Artificial Intelligence (AAAI'18)*. AAAI Press.

[16] Hertel, L., Baldi, P., and Gillen, D. (2020). Quantity vs. quality: On hyperparameter optimization for deep reinforcement learning. *CoRR*, abs/2007.14604.

[17] Islam, R., Henderson, P., Gomrokchi, M., and Precup, D. (2017). Reproducibility of benchmarked deep reinforcement learning tasks for continuous control. *CoRR*, abs/1708.04133.

[18] J. Parker-Holder, V., Nguyen, S. J., and Roberts (2020). Provably efficient online hyperparameter optimization with population-based bandits. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H., editors, *Proceedings of the 33rd International Conference on Advances in Neural Information Processing Systems (NeurIPS'20)*, volume 33, pages 17200–17211.

[19] Jaderberg, M., Dalibard, V., Osindero, S., Czarnecki, W., Donahue, J., Razavi, A., Vinyals, O., Green, T., Dunning, I., Simonyan, K., Fernando, C., and Kavukcuoglu, K. (2017). Population based training of neural networks. *arXiv:1711.09846 [cs.LG].*

[20] Lee, J., Hwangbo, J., Wellhausen, L., Koltun, V., and Hutter, M. (2020). Learning quadrupedal locomotion over challenging terrain. *Science in Robotics*, 5.

[21] Li, X., Zhang, J., Bian, J., Tong, Y., and Liu, T. (2019). A cooperative multi-agent reinforcement learning framework for resource balancing in complex logistics network. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS*, pages 980–988. International Foundation for Autonomous Agents and Multiagent Systems.

[22] Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2016). Continuous control with deep reinforcement learning. In Bengio, Y. and LeCun, Y., editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*.

[23] Lu, M., Shahn, Z., Sow, D., Doshi-Velez, F., and Lehman, L. H. (2020). Is deep reinforcement learning ready for practical applications in healthcare? A sensitivity analysis of duel-ddqn for hemodynamic management in sepsis patients. In *AMIA 2020, American Medical Informatics Association Annual Symposium, Virtual Event, USA, November 14-18, 2020*. AMIA.

[24] Meng, T. and Khushi, M. (2019). Reinforcement learning in financial markets. *Data*, 4(3):110.

[25] Modi, A., Jiang, N., Singh, S. P., and Tewari, A. (2018). Markov decision processes with continuous side information. In *Algorithmic Learning Theory (ALT'18)*, volume 83, pages 597–618.

[26] Obando-Ceron, J. and Castro, P. (2021). Revisiting rainbow: Promoting more insightful and inclusive deep reinforcement learning research. In *Proceedings of the 38th International Conference on Machine Learning, ICML*, volume 139 of *Proceedings of Machine Learning Research*, pages 1373–1383. PMLR.

[27] Raffin, A., Hill, A., Ernestus, M., Gleave, A., Kanervisto, A., and Dormann, N. (2019). Stable baselines3. `https://github.com/DLR-RM/stable-baselines3`.

[28] Ray, A., Achiam, J., and Amodei, D. (2019). Benchmarking Safe Exploration in Deep Reinforcement Learning.

[29] Runge, F., Stoll, D., Falkner, S., and Hutter, F. (2019). Learning to Design RNA. In *Proceedings of the International Conference on Learning Representations (ICLR'19)*. Published online: `iclr.cc`.

[30] Schubert, F., Awiszus, M., and Rosenhahn, B. (2021). Toad-gan: a flexible framework for few-shot level generation in token-based games. *IEEE Transactions on Games*, pages 1–1.

[31] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017a). Proximal policy optimization algorithms. *arXiv:1707.06347 [cs.LG].*

[32] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017b). Proximal policy optimization algorithms. *CoRR*, abs/1707.06347.

[33] Silver, D., Huang, A., Maddison, C., Guez, A., Sifre, L., Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. (2016). Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489.

[34] Wang, J., Kurth-Nelson, Z., Tirumala, D., Soyer, H., Leibo, J., Munos, R., Blundell, C., Kumaran, D., and Botvinick, M. (2016). Learning to reinforcement learn. *CoRR*, abs/1611.05763.

[35] Xu, L., Hoos, H., and Leyton-Brown, K. (2010). Hydra: Automatically configuring algorithms for portfolio-based selection. In Fox, M. and Poole, D., editors, *Proceedings of the Twenty-fourth National Conference on Artificial Intelligence (AAAI'10)*, pages 210–216. AAAI Press.

[36] Zhou, Z., Li, X., and Zare, R. (2017). Optimizing chemical reactions with deep reinforcement learning. *ACS central science*, 3(12):1337–1344.

## A Hardware, Software and Hyperparameters

**Hardware** All experiments are executed on a slurm GPU cluster consisting of six nodes with six to eight Nvidia RTX 2080 Ti GPUs each.

**Software** Our implementation uses the agents from stablebaselines3 [27].

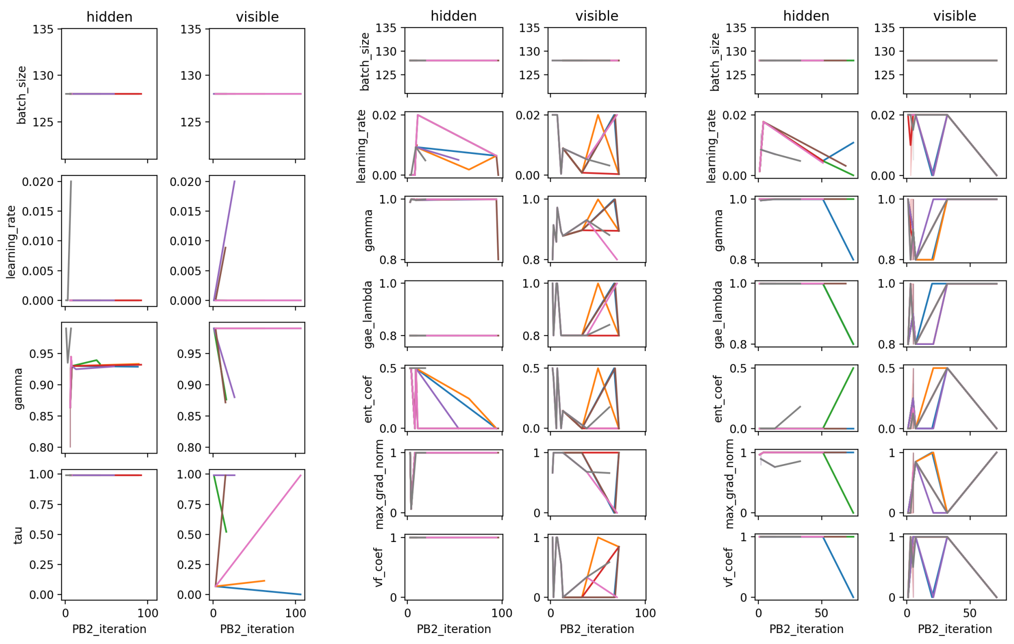
For CARLPendulum [5] we tune the hyperparameters of the DDPG agent [22], for CARLAcrobot and CARLLunarLander we use the PPO agent [32]. For tuning we use PB2 [18] as implemented in the ray tune package [28].

Our experiments can be reproduced via the scripts we provide at `https://github.com/automl-private/cRL_HPO`.

**PB2 Usage** PB2 [18] runs with 8 workers with a total timelimit of $24\,\mathrm{h}$ and a memory limit of $150\,\mathrm{GB}$. After $4096$ environment steps the hyperparameter configurations are adjusted by PB2. We start the optimization with a batch size of $128$, a learning rate of $0.00003$ and discount factor of $0.99$ for both algorithms. All other hyperparameters start at their default values.

In both cases, the learning rate is limited to be between $0.00001$ and $0.02$ and the discount factor between $0.8$ and $0.999$. For DDPG [22], $\tau$ can lie between $0.0$ and $0.99$. In PPO [32], the maximum gradient normalization and value function coefficient are both limited to between $0.0$ and $1.0$, the entropy coefficient to between $0.0$ and $0.5$ and the GAE parameter is between $0.8$ and $0.999$.

The schedules found by PB2 [18] are visualised in Figure 4. Please note that not all workers finished the desired number of timesteps due to the memory and time limits.



(a) CARLPendulum (b) CARLAcrobot (c) CARLLunarLander

Figure 4: Hyperparameter schedules found by PB2 [18] for hidden and visible context. The different linecolors indicate schedules found by individual PB2 workers (8 in total).

## B Context Training Distribution

During training only one context feature is varied in each environment. For training we use a set of 100 contexts. We sample the context feature from a Gaussian distribution, see Table 1 for details. The default value of the context feature is used as the mean $\mu$.

Table 1: Sampling context features from a Gaussian distribution $\mathcal{N}(\mu, \sigma)$.

| Environment | Context Feature | $\mu$ | $\sigma$ |
|---|---|---|---|
| CARLPendulum | gravity (g) | 10 | $0.1 \cdot 10$ |
| CARLAcrobot | link_length_1 | 1 | $0.1 \cdot 1$ |
| CARLLunarLander | gravity (GRAVITY_Y) | $-10$ | $0.1 \cdot 10$ |

## C Policy Sizes on CARLLunarLander

We identify the policy size as a possible cause for the poor results on CARLLunarLander. As PB2 cannot currently optimize discrete hyperparameters, we were not able to tune it in addition to our other hyperparameters. The default policy network has two hidden layers with $64$ units each. Rerunning the found hyperparameter policies with a greater amount of units (see Figure 5) shows a slightly more positive trend than the experiment in the main paper does. This suggests that the architecture should be adapted in CARLLunarLander, even with only smaller variations in the gravity as used here. It is unclear, however, if this would solve the problem entirely, as overall performance is only increased by a small margin. A reason might be that the network size is correlated to the other hyperparameters.

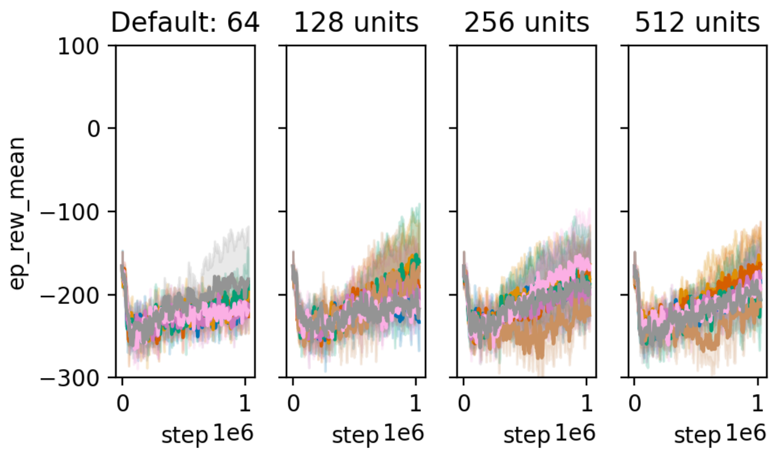


Figure 5: CARLLunarLander with increased number of policy units.