A Dataset Description

Our dataset is hosted on HuggingFace for easy and continued access: (redacted for peer-review). See the page there for in-depth information on data value and distributions. The croissant meta-data can also be found here: (redacted for peer-review). Everything needed to reproduce the data can be found in the 'experiments' branch of our GitHub repository: `https://anonymous.4open.science/r/arlbench-7043/tree/experiments` It is intended to be used in continued research on AutoRL, e.g. by using it in warmstarting optimizers, proposing novel analysis methods or meta-learning on it. The dataset is in the csv format, making it easily readable. We license it under the BDS-3 license.

B Reproducing Our Results

We provide code and runscripts for all of our dependencies in the 'experiments' branch of our repository: `https://anonymous.4open.science/r/arlbench-7043/arlbench/tree/experiments`

All scripts relating to the dataset creation and optimizer runs are in 'runscripts'. For the performance over time plots of the optimizers, see 'runtime_comparison'. 'rs_data_analysis' contains the analysis of the HPO landscapes. The subset selection and validation have directories with those same titles. Additionally, we provide all of our raw data in 'results_finished' with 'results_combined' containing aggregates of the dataset. Instructions for the usage of all of these can be found in the ReadMe file of that branch.

C Maintenance Plan

Following [Eggensperger et al., 2021] and [Pfisterer et al., 2022], we provide a maintenance plan for the future of ARLBench. For our feature roadmap, see:

Who Maintains ARLBench is being developed and maintained as a cooperation between (redacted) and (redacted).

Contact Improvement requests, issues and questions can be asked via issue in our GitHub repository: `https://anonymous.4open.science/r/arlbench-7043`. The contact e-mails we provide can be used for the same purpose.

Erratum There is no erratum.

Library Updates We plan on updating the library with new features, specifically more extensive state features, more algorithms and added environment frameworks. We also welcome updates via external pull requests which we will test and integrate into ARLBench. Changes will be communicated via the changelog of our GitHub and PyPI releases.

Support for Older Versions Older versions of ARLBench will continue to be available on PyPI and GitHub, but we will only provide limited support.

Contributions Contributions to ARLBench from external parties are welcome in any form, be extensions to other environment frameworks, added algorithms or extensions of the core interface. We describe the contribution process in our documentation: `https://anonymous.4open.science/r/arlbench-7043/main/CONTRIBUTING.html`. These contributions are managed via GitHub pull requests.

Dependencies All of our dependencies are listed here in the GitHub repository: `https://anonymous.4open.science/r/arlbench-7043/blob/main/pyproject.toml`.

D Overview of all Environments

Tables 2, 3 and 4 provide an overview of all environments we executed for a given RL algorithm, including the underlying framework used and the number of environment steps for training.

Category	Framework	Name	#timesteps
Atari	Envpool	BattleZone-v5	10M
Atari	Envpool	DoubleDunk-v5	10M
Atari	Envpool	Phoenix-v5	10M
Atari	Envpool	Qbert-v5	10M
Atari	Envpool	NameThisGame-v5	10M
Box2D	Envpool	LunarLander-v2	1M
Box2D	Envpool	LunarLanderContinuous-v2	1M
Walker	Brax	Ant	50M
Walker	Brax	HalfCheetah	50M
Walker	Brax	Hopper	50M
Walker	Brax	Humanoid	50M
Classic Control	Gymnax	Acrobot-v1	1M
Classic Control	Gymnax	CartPole-v1	100k
Classic Control	Gymnax	MountainCarContinuous-v0	20k
Classic Control	Gymnax	MountainCar-v0	1M
Classic Control	Gymnax	Pendulum-v1	100k
xland	XLand-xland	xland-DoorKey-5x5	1M
xland	XLand-xland	xland-EmptyRandom-5x5	100k
xland	XLand-xland	xland-FourRooms	1M
xland	XLand-xland	xland-Unlock	1M

Table 2: Environments for PPO.

Category	Framework	Name	#timesteps
Atari	Envpool	BattleZone-v5	10M
Atari	Envpool	DoubleDunk-v5	10M
Atari	Envpool	Phoenix-v5	10M
Atari	Envpool	Qbert-v5	10M
Atari	Envpool	NameThisGame-v5	10M
Box2D	Envpool	LunarLander-v2	1M
Classic Control	Gymnax	Acrobot-v1	100k
Classic Control	Gymnax	CartPole-v1	50k
Classic Control	Gymnax	MountainCar-v0	120k
xland	XLand-xland	xland-DoorKey-5x5	1M
xland	XLand-xland	xland-EmptyRandom-5x5	100k
xland	XLand-xland	xland-FourRooms	1M
xland	XLand-xland	xland-Unlock	1M

Table 3: Environments for DQN.

Category	Framework	Name	#timesteps
Box2D	Envpool	LunarLanderContinuous-v2	500k
Box2D	Envpool	BipedalWalker-v2	500k
Walker	Brax	Ant	5M
Walker	Brax	HalfCheetah	5M
Walker	Brax	Hopper	5M
Walker	Brax	Humanoid	5M
Classic Control	Gymnax	MountainCarContinuous-v0	50k
Classic Control	Gymnax	Pendulum-v1	20k

Table 4: Environments for SAC.

E Performance Comparisons with Other RL Frameworks

To validate the correctness of our implementations beyond unit testing, we compare their performance on a range of environments to established RL frameworks in terms of reward achieved and runtime.

Table 5 show the speedups we achieve in terms of runtime over StableBaselines3 (SB3) [Raffin et al., 2021] on all subsets while Tables 6, 7 and 8 list the same for each environment individually. As already discussed, we see a consistently large speedup, most pronounced for the Brax walkers with a factor of 8.57 for PPO and 10.67 for SAC. The lowest speedups we observe are still close to a factor of 2: 1.89 for DQN CartPole as well as 1.91 and 1.97 respectively for PPO LunarLander and LunarLanderContinuous.

Figure 8 compares the resulting learning curves between ARLBench, SB3 and the Brax default agent. We use the Brax agent instead of SB3 since SB3 performed significantly worse than we expected. In most of our tests we observed very similar behavior with the other framworks and ARLBench outperforming the other two times and showing comparable learning curves for all other experiments. In the case of DQN, where SB3 performed better on CartPole and worse on Pong, SB3's results look noisy, possibly causing this discrepancy in both directions. SB3 also outperforms ARLbench on Pendulum, though this difference is fairly slight. For PPO on Ant, ARLBench performs quite a bit better than the Brax default agent, though their performances of SAC are the same. Overall this shows that our algorithms perform on par with other commonly used implementations.

Algorithm	Set	ARLBench	SB3	Speedup
PPO	All	1.97h	7.12h	3.61h
PPO	Subset	0.44h	1.70h	3.86
DQN	All	3.87h	11.10h	2.87
DQN	Subset	0.83h	2.45h	2.95
SAC	All	1.25h	7.23h	5.78
SAC	Subset	0.37h	2.19h	5.92
Sum	All	7.09h	25.45h	3.59
Sum	Subset	1.64h	6.34h	3.87

Table 5: Runtime comparisons for a single RL training between ARLBench and StableBaselines3 (SB3) [Raffin et al., 2021] on the set of all environments and the selected subset. The numbers are based on the results in Tables 6, 7, and 8. For each environment category, we use the runtimes from the experiments to estimate the overall runtime for this category.

Category	Framework	Name	ARLBench	SB3	Speedup
Classic Control	Envpool	CartPole-v1	5.42s	25.54s	4.72
Classic Control	Envpool	LunarLander-v2	125.87s	248.54s	1.97
Classic Control	Envpool	LunarLanderContinuous-v2	162.77s	311.25s	1.91
Classic Control	Envpool	Pendulum-v1	5.98s	21.87s	3.66
Atari	Envpool	Pong-v5	1161.11s	3728.58s	3.21
Walker	Envpool	Ant	194.09s	1048.84s	
Walker	Brax	Ant	122.28s	1048.84s*	8.57
				Average	4.00

Table 6: Speedup of ARLBench PPO compared to StableBaselines3 (SB3) [Raffin et al., 2021] on different envrionments. *Note: Since SB3 is not compatible with Brax without manual interface adapation, we compare the results of MuJoCo + SB3 and Brax + ARLBench.

Category	Framework	Name	ARLBench	SB3	Speedup
Classic Control	Envpool	CartPole-v1	21.5s	40.68s	1.89
Classic Control	Envpool	LunarLander-v2	95.27s	194.61s	2.04
Atari	Envpool	Pong-v5	2602.69s	7373.40s	2.83
				Average	2.25

Table 7: Speedup of ARLBench DQN compared to StableBaselines3 (SB3) [Raffin et al., 2021] on different envrionments.

Category	Framework	Name	ARLBench	SB3	Speedup
Classic Control	Envpool	Pendulum-v1	17.32s	105.67s	6.10
Classic Control	Envpool	LunarLanderContinuous-v2	365.45s	2425.04s	6.64
Walker	Envpool	Ant	930.06s	5245.17s	
Walker	Brax	Ant	491.70s	5245.17s*	10.67
				Average	7.80

Table 8: Speedup of ARLBench SAC compared to StableBaselines3 (SB3) [Raffin et al., 2021] on different envrionments. *Note: Since SB3 is not compatible with Brax without manual interface adapation, we compare the results of MuJoCo + SB3 and Brax + ARLBench.

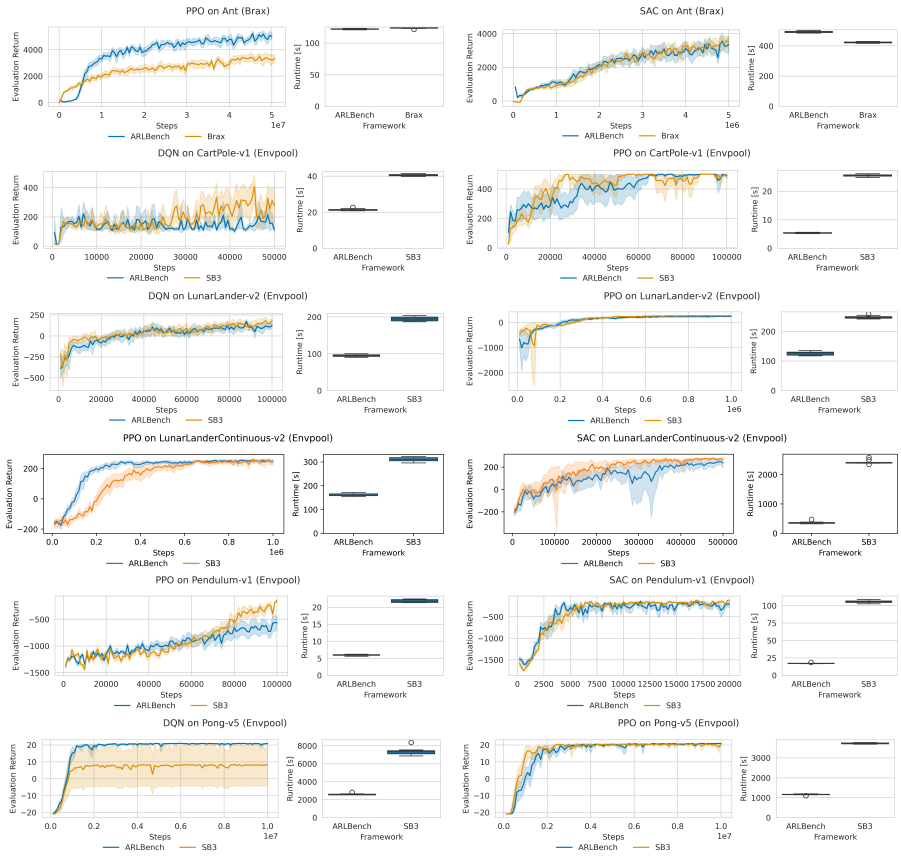

Figure 8: Performance comparisons of ARLBench and the Brax default agent, StableBaselines3 [Raffin et al., 2021]

F Algorithm Search Spaces

For all algorithms, we used extensive search spaces covering almost all hyperparameters that are commonly optimized. The search spaces for PPO, DQN and SAC are presented in Table 9, 10 and 11 respectively. We choose not to optimize some hyperparameters to keep the computational resources constant for each training. The default values for these hyperparameters for each environment domain have been inferred from stable-baselines3 zoo Raffin [2020] and Google Brax's hyperparameter sweeps Freeman et al. [2021b] and are shown in Table 9, 10 and 11 accordingly. The search space for the batch sizes was set to one power of two below and above its baseline value.

Hyperparameter	Box2D	XLand	Atari	CC	Brax
batch size	$\{32, 64, 128\}$		$\{128, 256, 512\}$		$\{512, 1024, 2048\}$
number of environments	16	8			2048
number of steps	1024	32	128	32	512
update epochs	4	10	4		
learning rate	$\log([10^{-6}, 10^{-1}])$				
entropy coefficient	$[0.0, 0.5]$				
gae lambda	$[0.8, 0.9999]$				
policy clipping	$[0.0, 0.5]$				
value clipping	$[0.0, 0.5]$				
normalize advantages	$\{\text{Yes}, \text{No}\}$				
value function coefficient	$[0.0, 1.0]$				
max gradient norm	$[0.0, 1.0]$				

Table 9: The hyperparameter search space for PPO. To keep the computational costs feasible we choose not to optimize the number of steps per epoch and update epochs.

Hyperparameter	Atari	Box2D	CC	XLand
batch size	$\{16, 32, 64\}$	$\{64, 128, 256\}$		$\{32, 64, 128\}$
number of environments	8	4	1	4
buffer priority sampling	$\{\text{Yes}, \text{No}\}$			
buffer α	$[0.01, 1.0]$			
buffer β	$[0.01, 1.0]$			
buffer ϵ	$\log([10^{-7}, 10^{-3}])$			
buffer size	$[1024, 1M]$			
initial epsilon	$[0.5, 1.0]$			
target epsilon	$[0.001, 0.2]$			
learning rate	$\log([10^{-6}, 10^{-1}])$			
learning starts	$[1, 2048]$			
use target network	$\{\text{Yes}, \text{No}\}$			
target update interval	$[1, 2000]$			

Table 10: The hyperparameter search space for DQN. The target update interval is a conditional hyperparameter that is only optimized when a target network is used. Similarly, buffer α, β and ϵ are only optimized when priority sampling is used. If the number of training steps is smaller than the upper limit of the buffer size, the buffer size limit is reduced accordingly.

Hyperparameter	Box2D	CC	Brax
batch size	$\{128, 256, 512\}$	$\{256, 512, 1024\}$	$\{512, 1024, 2048\}$
number of environments	1	1	64
buffer priority sampling	$\{\text{Yes}, \text{No}\}$		
buffer α	$[0.01, 1.0]$		
buffer β	$[0.01, 1.0]$		
buffer ϵ	$\log([10^{-7}, 10^{-3}])$		
buffer size	$[1024, 1M]$		
learning rate	$\log([10^{-6}, 10^{-1}])$		
learning starts	$[1, 2048]$		
use target network	$\{\text{Yes}, \text{No}\}$		
tau	$[0.01, 1.0]$		
reward scale	$\log([0.1, 10])$		

Table 11: The hyperparameter search space for SAC. The hyperparameter tau is a conditional parameter that is only optimized when a target network is used. Similarly, buffer α, β and ϵ are only optimized when priority sampling is used. If the number of training steps is smaller than the upper limit of the buffer size, the buffer size limit is reduced accordingly.

G Subset Selection

We provide additional information on the subset selection in the form of alternative selection methods and a more detailed look into the results, including environment weights.

G.1 Alternative Methods

In addition to our chosen method of rank-based normalization in combination with the Spearman correlation as a distance metric, we compare alternative normalization methods as well as MSE for the distance. Figure 9 shows the validation error of different combinations while Figure 10 shows the resulting Spearman correlation to the full environment set. While MSE might produce a good validation error, the resulting correlation is significantly worse than using the Spearman correlation for the distance. Min-max normalization performs slightly worse than rank normalization for the validation error. Therefore we chose rank-based normalization with Spearman correlation for our subset selection.

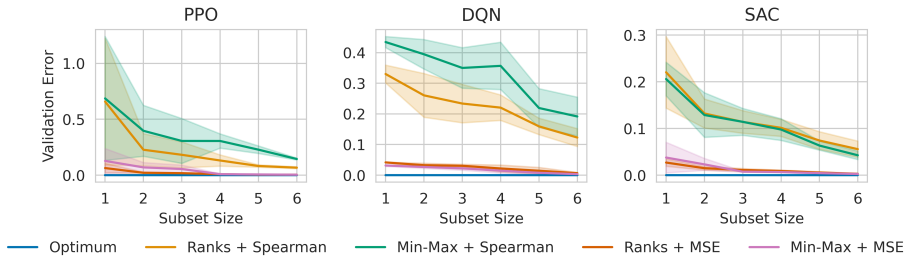

Figure 9: Comparison of the validation error of different ranking methods and error functions based on the subset size.

Algorithm	Environments (with predicted weights)	ρ_s
PPO	$0.31\times$ LunarLander, $0.19\times$ HalfCheetah, $0.19\times$ BattleZone, $0.16\times$ xland-EmptyRandom, $0.13\times$ xland-FourRooms	0.97
DQN	$0.31\times$ Acrobot, $0.30\times$ xland-DoorKey, $0.23\times$ BattleZone, $0.14\times$ xland-FourRooms	0.93
SAC	$0.36\times$ BipedalWalker, $0.32\times$ HalfCheetah, $0.18\times$ MountainCarContinuous, $0.14\times$ Pendulum	0.96

Table 12: The environment subsets selected for each algorithm with their Spearman correlation to the full environment set.

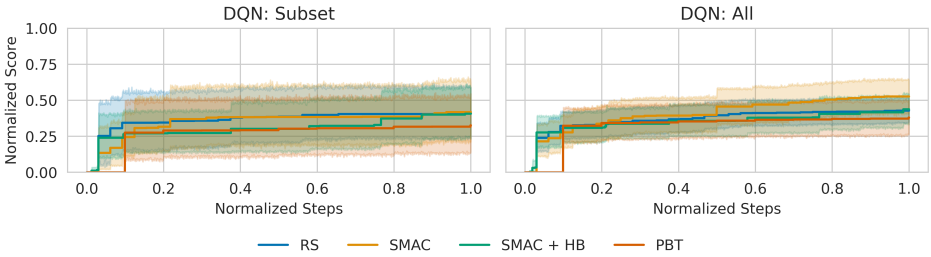

Figure 11: Anytime performance of the HPO methods for DQN.

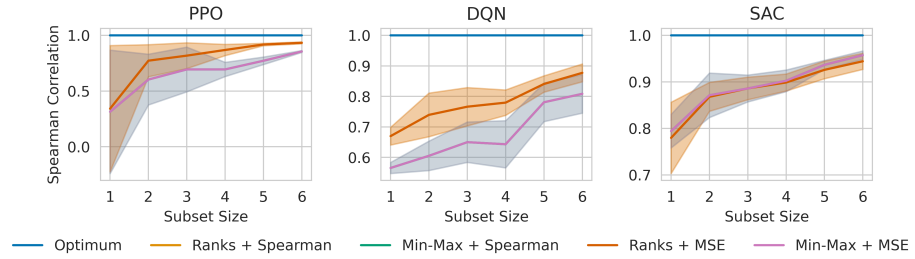

Figure 10: Comparison of the validation error of different ranking methods and error functions based on the subset size.

G.2 Extended Subset Results

In addition to the environments in the subsets, we also provide the exact weights for each environment in the subsets in Table 12. Furthermore, Figures 11 and 12 show the optimization-over-time results for DQN and SAC.

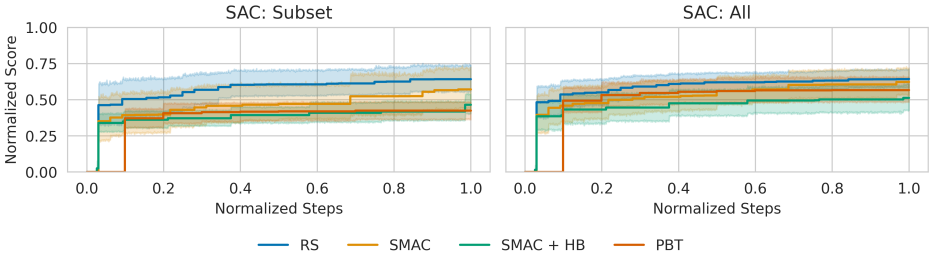

Figure 12: Anytime performance of the HPO methods for SAC.

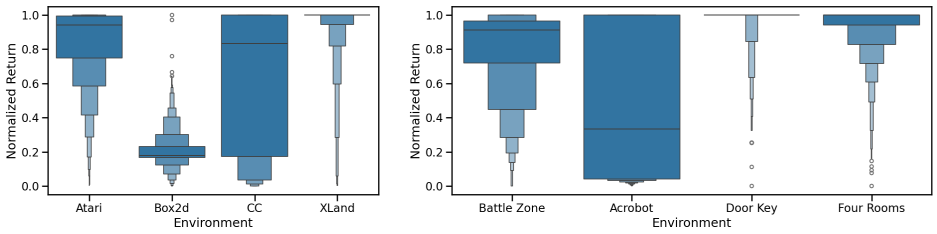

Figure 13: Score distribution across environment domains and the selected subset of DQN.

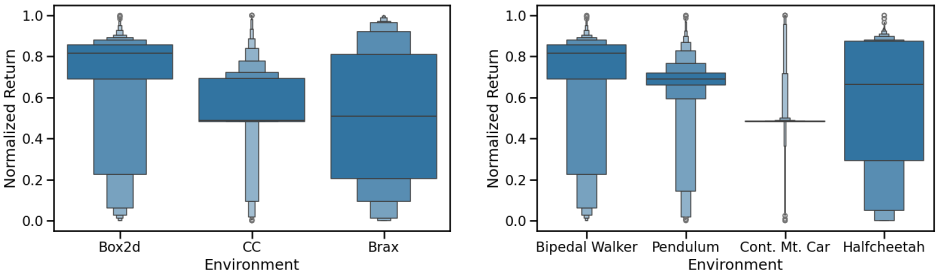

Figure 14: Score distribution across environment domains and the selected subset of SAC.

H Landscape Analysis

We use DeepCave [Sass et al., 2022] to analyze our performance dataset with regards to performance distribution, hyperparameter importance and budget correlation over time. Please note that in some cases, results can be missing due to consistent numerical errors in the analysis, e.g. in the case of SAC on Halfcheetah.

H.1 Performance Distribution

Completing the results from the performance distributions comparison in Section 4.3, Figures 13 and 14 show the distribution of scores for the domains and subsets of DQN and SAC, respectively. Just like for PPO, there are fairly direct correspondences between selected environments and the score distributions of the full domains. The only seeming exception is Box2D for DQN which has a lot of low scores that are not directly represented by one selected environment. Acrobot in that subset, however, covers a lot of such bad configurations even though it has higher performances overall.

H.2 Hyperparameter Importances

Tables 13, 14 and 15 show extended information on the number of important hyperparameters for each environment domain as well as the subset and full environment set. The set of top 3 interactions are shown in Table 16 for PPO, Table 17 for DQN and Table 18 for SAC. We also include the full set of importance plots for each environment in Figures 15, 16, 17 and 18 for PPO, Figures 19, 20 and 21 for DQN and Figure 22 for SAC.

	Atari	Box2D	CC	Xland	Brax	All	Subset
#HPs with over 5% importance	2.4	1.0	3.4	2.25	3	2.41	2.0
#HP interactions over 5%	0.8	0.0	0.8	1.75	3.33	1.34	1.0

Table 13: Fraction of hyperparameters with over 5% importance on the full set and subset for PPO.

	Atari	Box2D	CC	Xland	All	Subset
#HPs with over 5% importance	3.4	1.0	3.4	2.25	2.14	2.75
#HP interactions over 5%	1.2	0.0	0.67	1.75	0.8	0.75

Table 14: Fraction of hyperparameters with over 5% importance on the full set and subset for DQN.

	Box2D	CC	Brax	All	Subset
#HPs with over 5% importance	1.0	6.0	2.33	3.11	3.0
#HP interactions over 5%	1.0	1.0	2.0	1.33	0.67

Table 15: Fraction of hyperparameters with over 5% importance on the full set and subset for SAC.

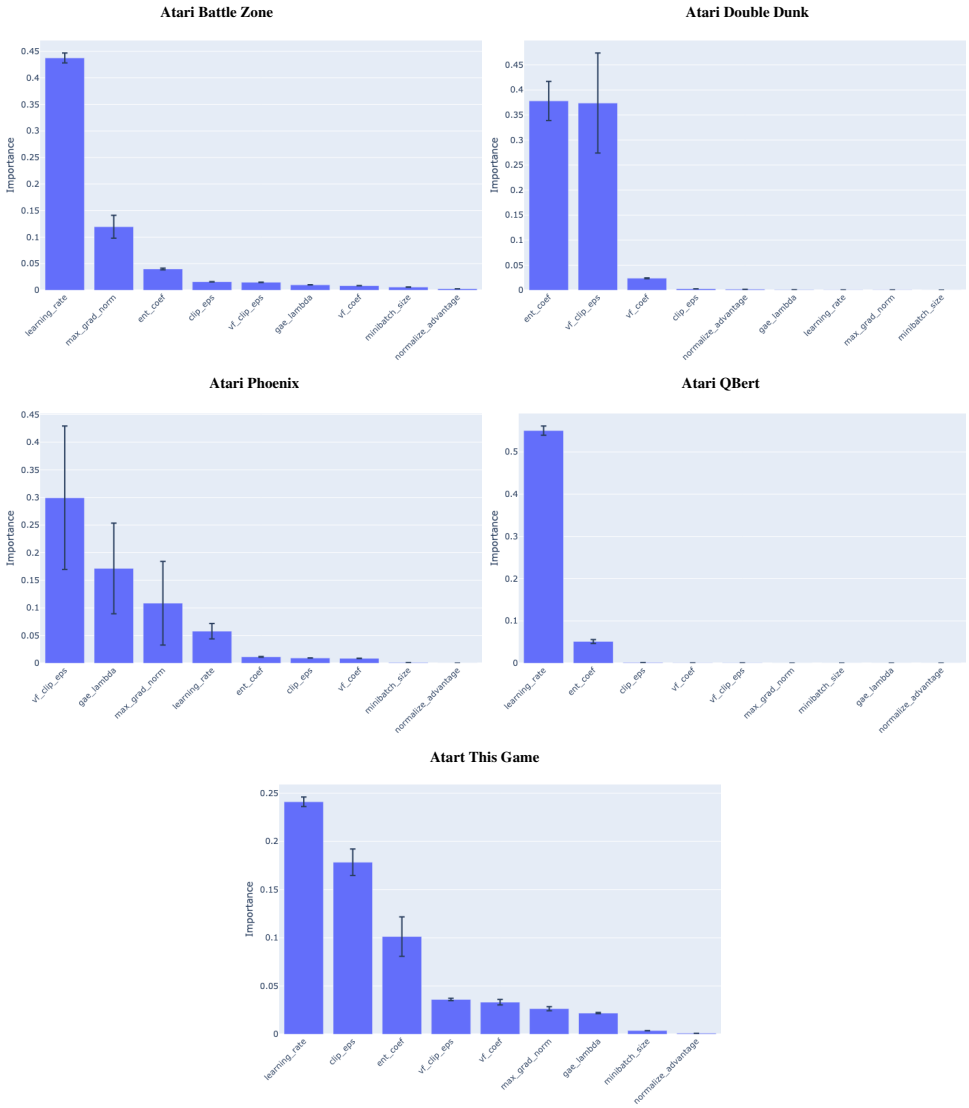

Figure 15: Hyperparameter Importances for PPO: Atari.

Environment	Highest Interaction	2nd Highest Interaction	3rd Highest Interaction
atari qbert	(clip eps, lr) : 0.13	(ent coef, lr) : 0.07	(lr, vf coef) : 0.02
atari double dunk	(ent coef, vf coef) : 0.13	(ent coef, vf clip eps) : 0.02	(clip eps, ent coef) : 0.02
atari phoenix	(lr, max grad norm) : 0.06	(clip eps, lr) : 0.05	(ent coef, gae lambda) : 0.03
atari this game	(clip eps, vf clip eps) : 0.03	(clip eps, lr) : 0.03	(clip eps, max grad norm) : 0.03
atari battle zone	(lr, vf coef) : 0.07	(lr, norm. advantage) : 0.04	(gae lambda, lr) : 0.03
box2d lunar lander	(gae lambda, lr) : 0.02	(ent coef, lr) : 0.02	(clip eps, lr) : 0.02
cc acrobot	(lr, minibatch size) : 0.06	(max grad norm, minibatch size) : 0.03	(lr, max grad norm) : 0.03
cc cartpole	(gae lambda, lr) : 0.06	(clip eps, lr) : 0.03	(ent coef, lr) : 0.03
cc mt car	(clip eps, ent coef) : 0.07	(lr, vf coef) : 0.05	(clip eps, lr) : 0.04
cc pendulum	(lr, max grad norm) : 0.04	(ent coef, lr) : 0.04	(ent coef, norm. advantage) : 0.03
cc cont. mt car	(clip eps, ent coef) : 0.04	(ent coef, norm. advantage) : 0.03	(clip eps, lr) : 0.03
xland doorkey	(lr, norm. advantage) : 0.21	(max grad norm, minibatch size) : 0.15	(ent coef, norm. advantage) : 0.04
xland empty	(lr, norm. advantage) : 0.06	(ent coef, lr) : 0.05	(learning rate, vf coef) : 0.03
xland fourrooms	(lr, vf clip eps) : 0.05	(clip eps, vf clip eps) : 0.03	(ent coef, lr) : 0.03
xland unlock	(clip eps, ent coef) : 0.05	(ent coef, max grad norm) : 0.05	(ent coef, vf clip eps) : 0.03
brax ant	(lr, max grad norm) : 0.07	(ent coef, lr) : 0.06	(clip eps, lr) : 0.05
brax hopper	(clip eps, lr) : 0.13	(lr, vf coef) : 0.12	(lr, max grad norm) : 0.09
brax humanoid	(ent coef, lr) : 0.19	(gae lambda, vf coef) : 0.08	(lr, vf coef) : 0.07

Table 16: Interactions effects for PPO via fANOVA.

Environment	Highest Interaction	2nd Highest Interaction	3rd Highest Interaction
atari double dunk	(alpha, beta): 0.05	(alpha, target network): 0.05	(alpha, target eps.): 0.04
atari qbert	(batch size, lr): 0.05	(lr, l. starts): 0.05	(buffer size, lr): 0.02
atari phoenix	(alpha, lr): 0.02	(buffer size, lr): 0.01	(beta, target epsilon): 0.01
atari this game	(initial eps., lr) : 0.02	(initial eps., lr) : 0.02	(initial eps., target network) : 0.02
atari battle zone	(initial eps., l. starts) : 0.02	(alpha, initial eps.) : 0.02	(alpha, target update) : 0.02
box2d lunar lander	(initial eps., lr) : 0.04	(alpha, lr) : 0.03	(lr, target update) : 0.03
cc acrobot	(lr, target update interval) : 0.05	(buffer size, lr) : 0.01	(initial eps., lr) : 0.01
cc cartpole	(alpha, lr) : 0.03	(buffer size, lr) : 0.02	(initial eps., lr) : 0.02
cc mountain car	(lr, target update) : 0.11	(alpha, lr) : 0.03	(alpha, target update) : 0.03
xland door key	(alpha, lr) : 0.05	(lr, l. starts) : 0.04	(alpha, initial eps.) : 0.02
xland empty random	(alpha, target eps.) : 0.12	(alpha, lr) : 0.07	(alpha, init. eps.) : 0.04
xland four rooms	(alpha, lr) : 0.05	(alpha, l. starts) : 0.04	(buffer eps., learning starts) : 0.03
xland unlock	(alpha, buffer size) : 0.06	(alpha, beta) : 0.06	(l. starts, target update) : 0.05

Table 17: Interaction effects for DQN hyperparameters via fANOVA.

Environment	Highest Interaction	2nd Highest Interaction	3rd Highest Interaction
box2d bipedal walker	(lr, reward scale) : 0.07	(buffer alpha, lr) : 0.03	(batch size, lr) : 0.02
cc pendulum	(batch size, lr) : 0.05	(buffer alpha, batch size) : 0.03	(buffer size, lr) : 0.02
cc cont. mt car	(alpha, alpha auto) : nan	(alpha, buffer alpha) : nan	(alpha, batch size) : nan
brax ant	(lr, tau) : 0.1	(lr, reward scale) : 0.08	(buffer size, lr) : 0.01
brax halfcheetah	(batch size, buffer size) : 0.04	(buffer alpha, buffer prio sampling) : 0.02	(lr, l. starts) : 0.02
brax hopper	(lr, tau) : 0.17	(lr, reward scale) : 0.05	(lr, target network) : 0.04
brax humanoid	(alpha, alpha auto) : nan	(alpha, buffer alpha) : nan	(alpha, batch size) : nan

Table 18: Interaction effects for SAC hyperparameters via fANOVA.

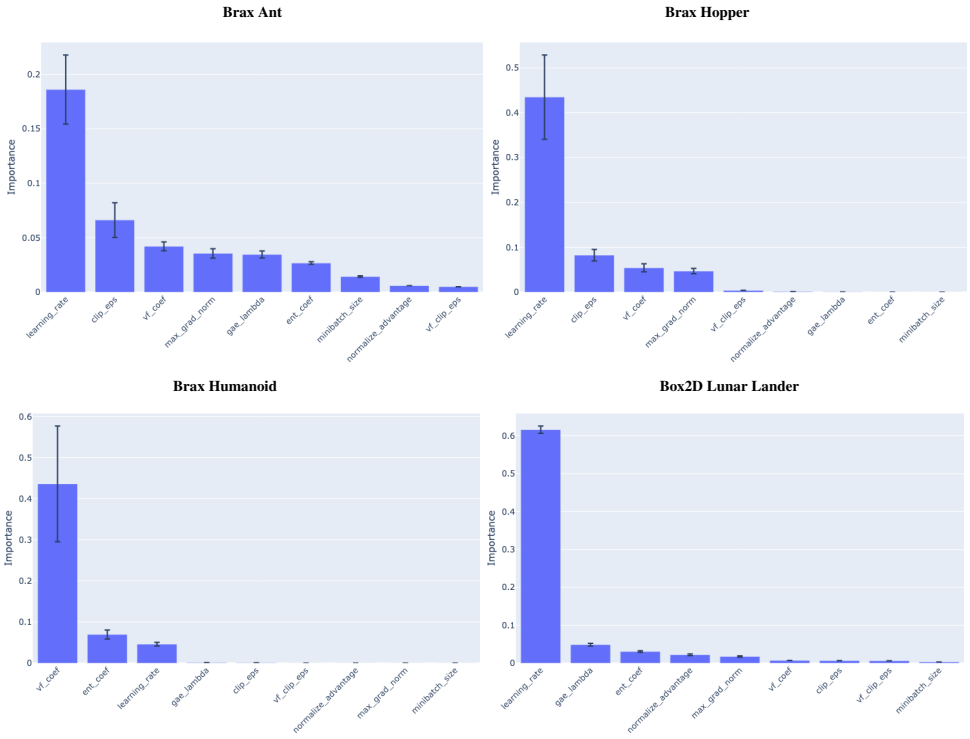

Figure 16: Hyperparameter Importances for PPO: Brax and Box2D.

H.3 Budget correlations

We show the budget correlation plots for all budgets (Figures 23, 24, 25 and 26 for PPO, Figures 27, 28 and 29 for DQN and Figure 30 for SAC). We see that most correlations are strong or very strong with some numerical inconsistencies in mountaincar and brax halfcheetah. XLand is the the only domain with a strong trend over time, for DQN most correlations only become strong after about 30-40% of training while the same is true for DoorKey of PPO. These are consistent in the domains, though: we see low or no correlations for Brax (likely due to numerical issues) and in Classic Control, strong correlations otherwise. Atari and Box2D are other domains with strong correlations while XLand tends to need a warmup phase.

I Resource Consumption

I.1 Hardware

To conduct the experiments detailed in this paper, we pooled various computing resources. Below, we describe the different hardware setups used for CPU and GPU-based training.

CPU Jobs Compute nodes with CPUs of type AMD Milan 7763, 2.45 GHz, each 2x 64 cores, 128GB main memory

GPU Jobs

V100 Cluster: Compute nodes with CPUs of type Intel Xeon Platinum 8160, 2.1 GHz, each 2x 24 cores, 180GB main memory. Each node comes with 16 GPUs of type NVIDIA V100-SXM2 with NVLink and 32 GB HBM2, 5120 CUDA cores, 640 Tensor cores, 128 GB main memory

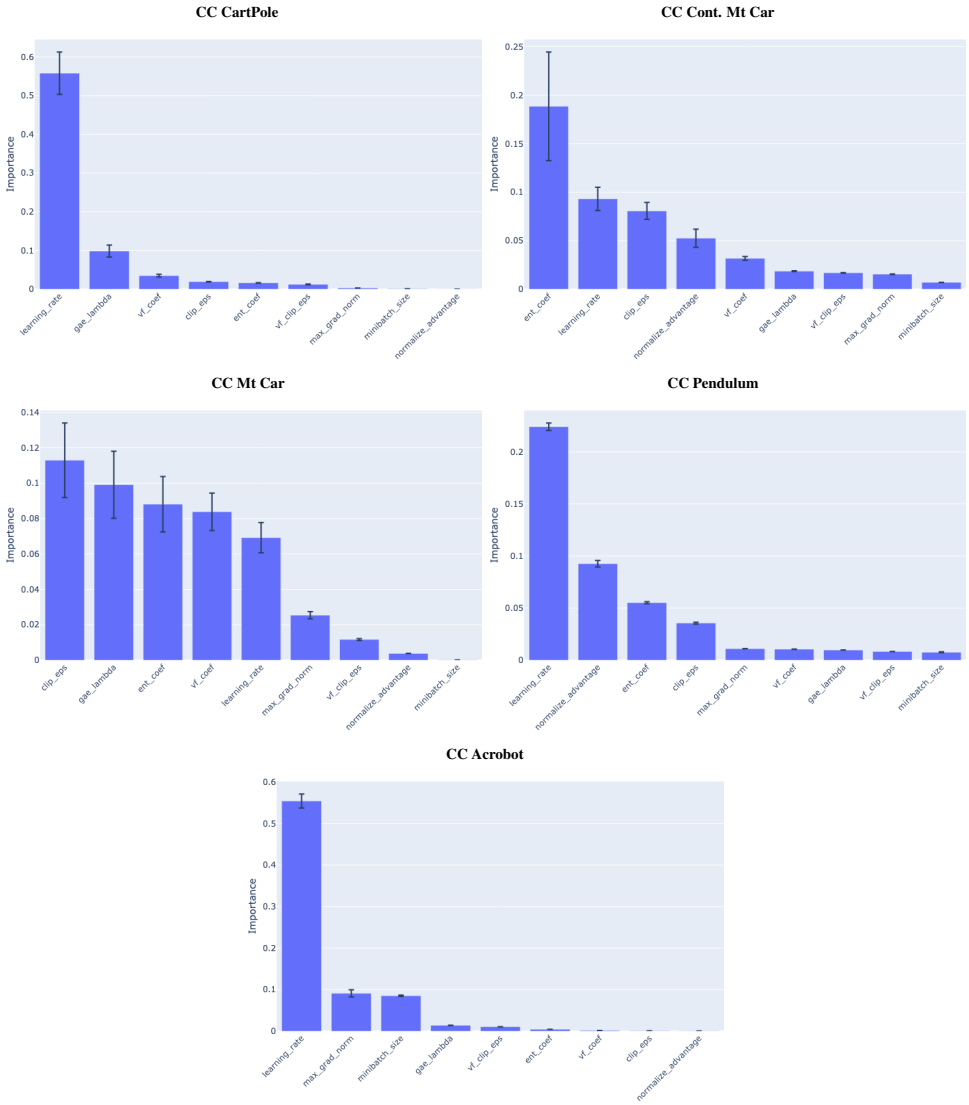

Figure 17: Hyperparameter Importances for PPO: Classic Control.

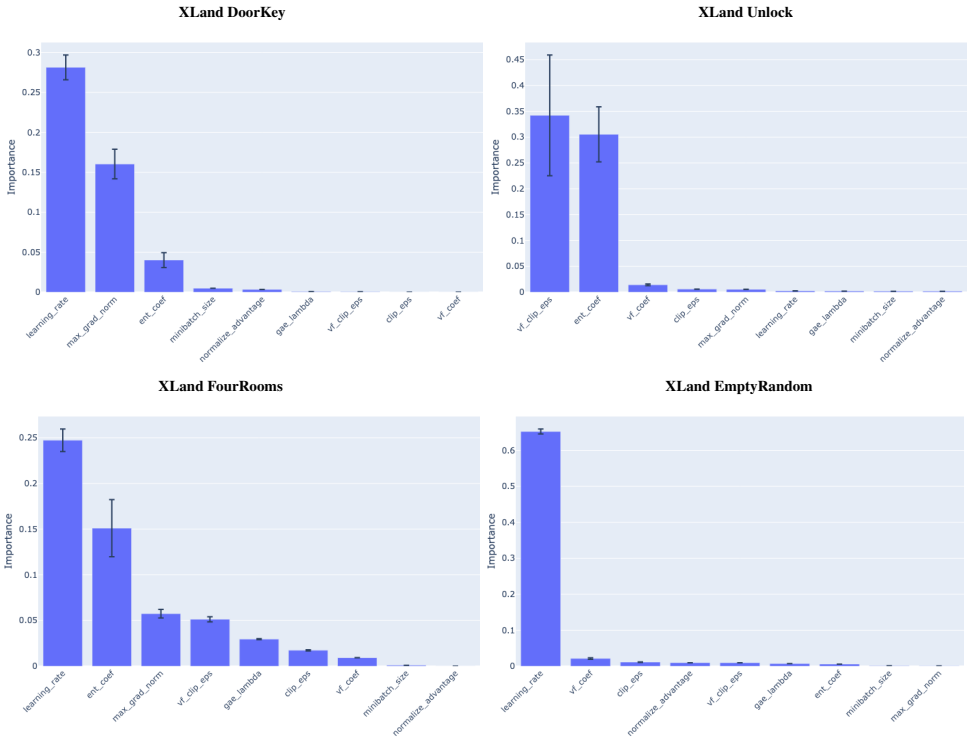

Figure 18: Hyperparameter Importances for PPO: XLand.

A100 Cluster: Compute nodes with CPUs of type AMD Milan 7763, 2.45 GHz, each 2x 64 cores, 126GB main memory. Each node comes with 1 GPU of type NVIDIA A100 with NVLink and 40 GB HBM2, 6,912 CUDA cores, 432 Tensor cores, 16 GB main memory

H100 Cluster: Compute nodes with CPUs of type Intel Xeon 8468 Sapphire, 2.1 GHz, each 2x 48 cores, 512GB main memory. Each node comes with 4 GPUs of type NVIDIA H100 with NVLink and 96 GB HBM2e, 16,896 CUDA cores, 528 Tensor cores, 512 GB main memory

All runtime experiments were run on the same setup with H100 GPUs.

I.2 Resource Consumption

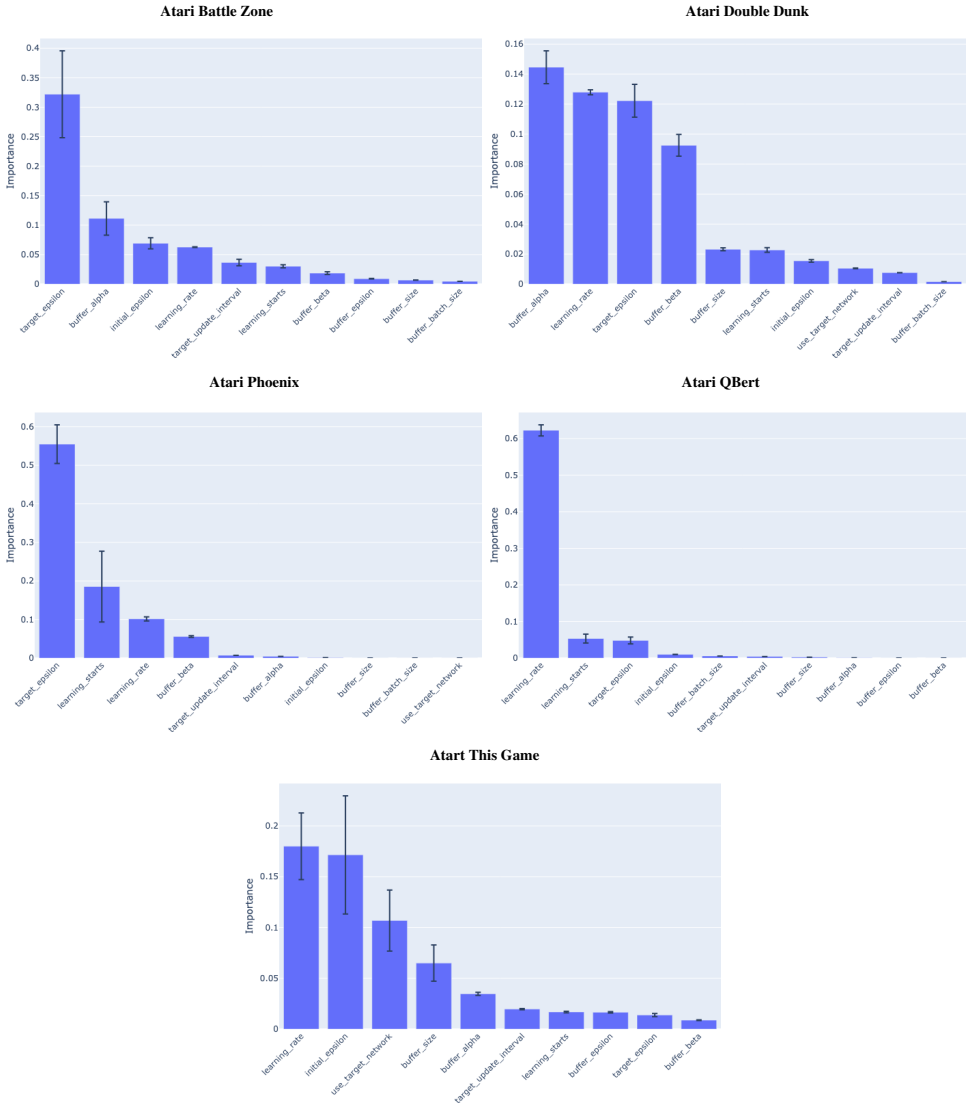

Figure 19: Hyperparameter Importances for DQN: Atari.

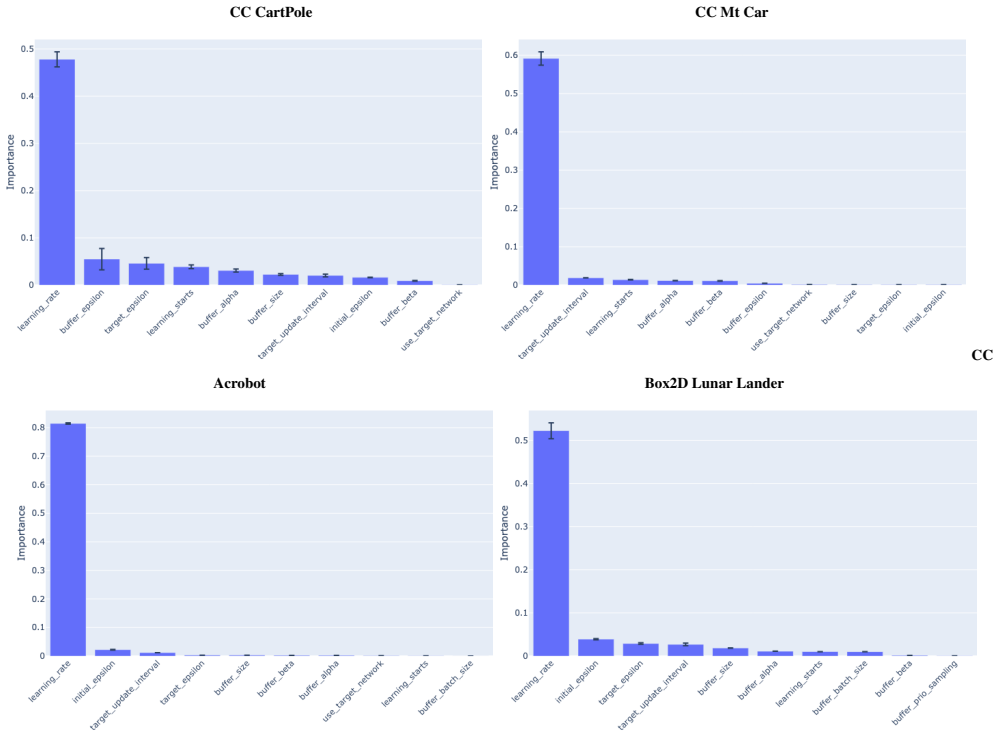

Figure 20: Hyperparameter Importances for DQN: Classic Control and Box2D.

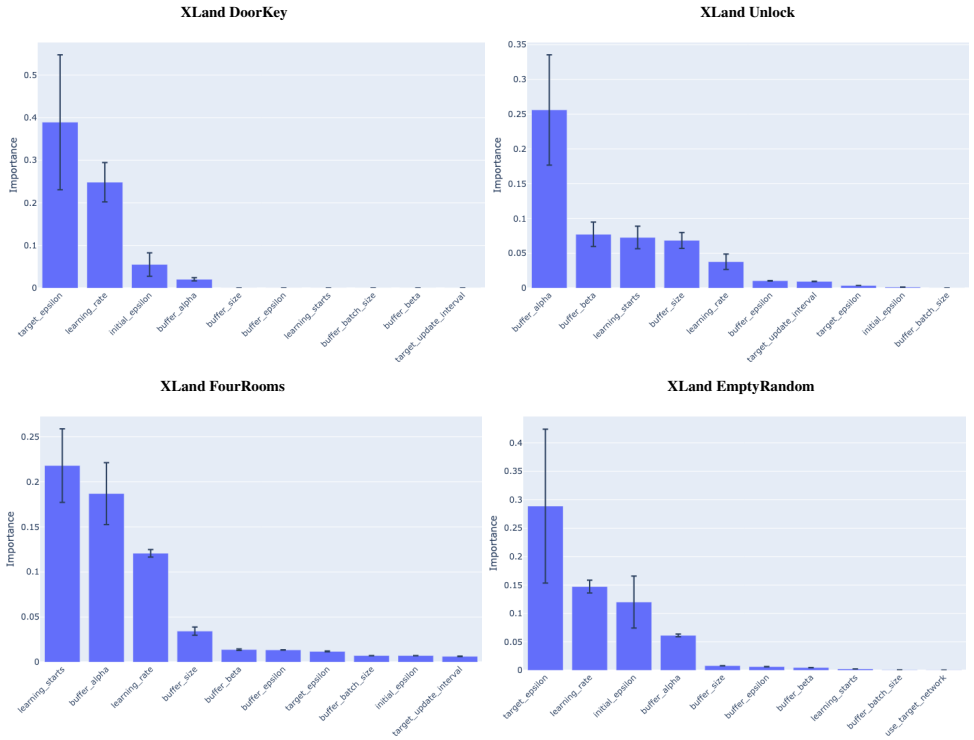

Figure 21: Hyperparameter Importances for DQN: XLand.

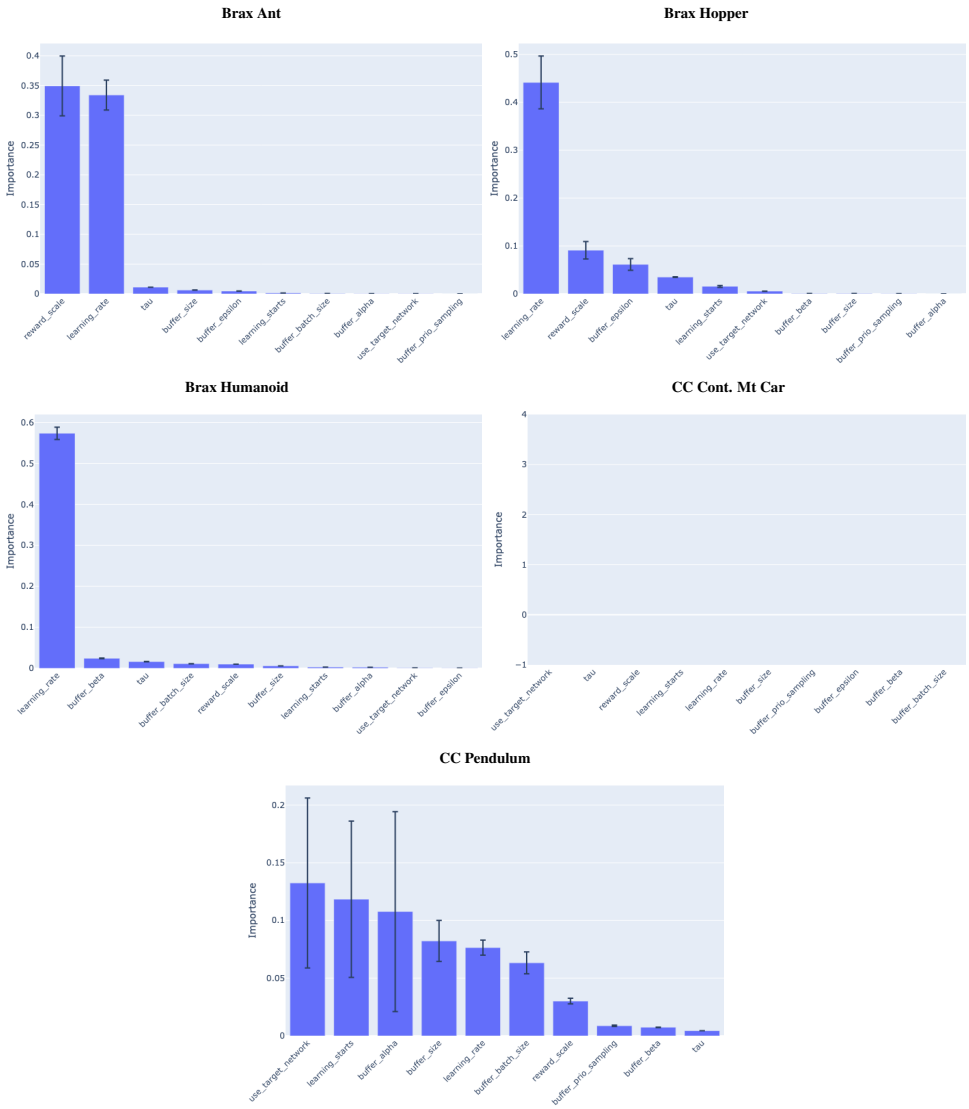

Figure 22: Hyperparameter Importances for SAC.

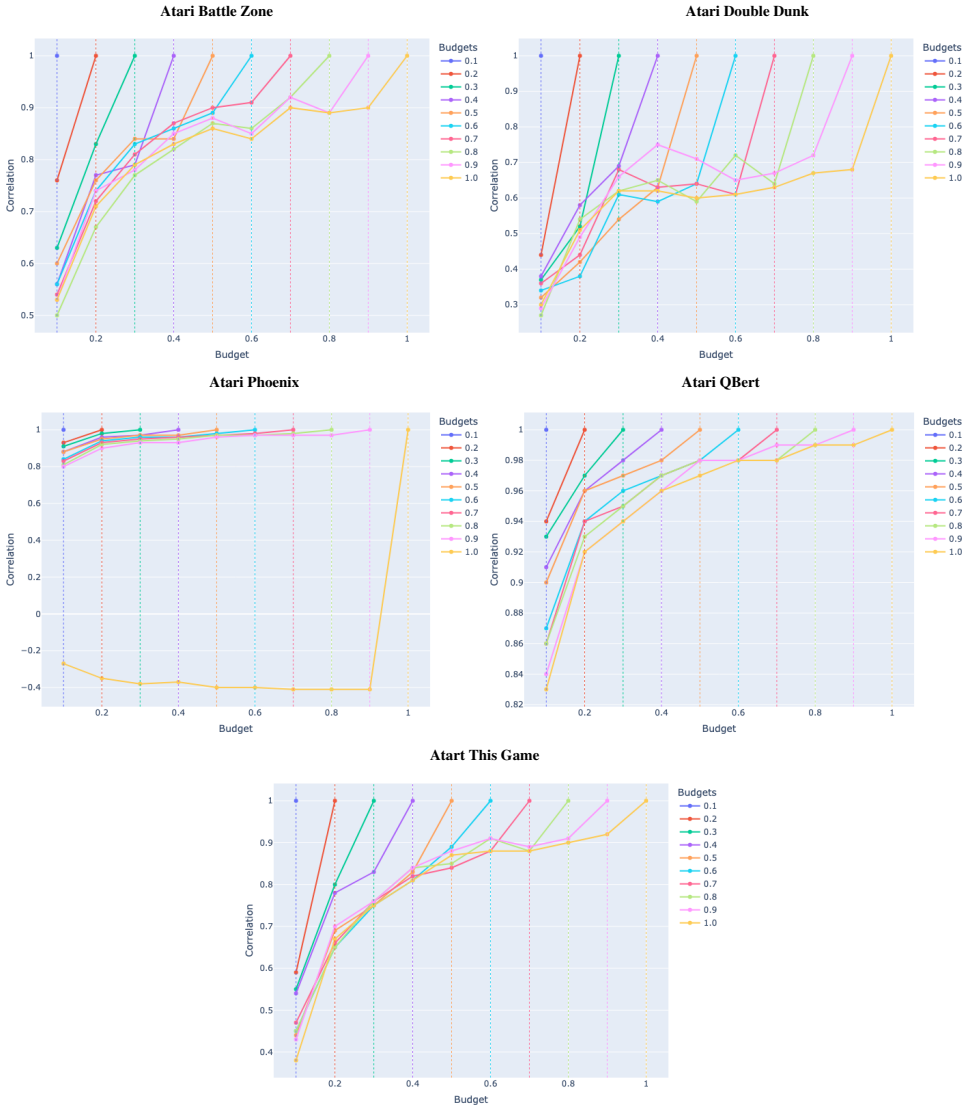

Figure 23: Budget Correlations for PPO: Atari.

Algorithm	Environment	Platform	Runtime [s]	#runs	Total runtime [h]
DQN	Acrobot-v1	CPU	26.103955	4096	29.700500
DQN	BattleZone-v5	GPU	2967.691667	2816	2321.394370
DQN	CartPole-v1	CPU	10.266920	4096	11.681474
DQN	DoubleDunk-v5	GPU	2918.080465	2816	2282.587386
DQN	LunarLander-v2	CPU	34.471462	4096	39.220863
DQN	MiniGrid-DoorKey-5x5	CPU	81.438681	4096	92.659121
DQN	MiniGrid-EmptyRandom-5x5	CPU	30.321538	4096	34.499172
DQN	MiniGrid-FourRooms	CPU	172.311806	4096	196.052543
DQN	MiniGrid-Unlock	CPU	94.675653	4096	107.719854
DQN	MountainCar-v0	CPU	19.401896	4096	22.075046
DQN	NameThisGame-v5	GPU	2970.152392	2816	2323.319205
DQN	Phoenix-v5	GPU	2710.294604	2816	2120.052668
DQN	Qbert-v5	GPU	2943.792347	4096	3349.381515
PPO	Acrobot-v1	CPU	15.342503	4096	17.456360
PPO	BattleZone-v5	GPU	1154.294962	2816	902.915170
PPO	CartPole-v1	CPU	7.947444	4096	9.042426
PPO	DoubleDunk-v5	GPU	1083.078698	2816	847.208226
PPO	LunarLander-v2	CPU	162.970049	4096	185.423700
PPO	LunarLanderContinuous-v2	CPU	300.469149	4096	341.867121
PPO	MiniGrid-DoorKey-5x5	CPU	81.226910	4096	92.418173
PPO	MiniGrid-EmptyRandom-5x5	CPU	26.369733	4096	30.002896
PPO	MiniGrid-FourRooms	CPU	179.836580	4096	204.614064
PPO	MiniGrid-Unlock	CPU	112.331167	4096	127.807905
PPO	MountainCar-v0	CPU	13.210608	4096	15.030736
PPO	MountainCarContinuous-v0	CPU	7.684076	4096	8.742771
PPO	NameThisGame-v5	GPU	1130.458910	2816	884.270080
PPO	Pendulum-v1	CPU	13.812708	4096	15.715793
PPO	Phoenix-v5	GPU	955.167698	2816	747.153399
PPO	Qbert-v5	GPU	1145.068274	4096	1302.833237
PPO	Brax Ant	GPU	220.871733	4096	251.302949
PPO	Brax HalfCheetah	GPU	802.293823	4096	912.832083
PPO	Brax Hopper	GPU	394.806557	4096	449.202127
PPO	Brax Humanoid	GPU	311.397193	4096	354.300806
SAC	BipedalWalker-v3	CPU	486.322458	4096	553.326886
SAC	LunarLanderContinuous-v2	CPU	381.216796	4096	433.739999
SAC	MountainCarContinuous-v0	CPU	557.128674	4096	633.888624
SAC	Pendulum-v1	CPU	111.763764	4096	127.162327
SAC	Brax Ant	GPU	943.786708	4096	1073.819544
SAC	Brax HalfCheetah	GPU	2472.490396	4096	2813.144628
SAC	Brax Hopper	GPU	1328.451953	4096	1511.483111
SAC	Brax Humanoid	GPU	947.802141	4096	1078.388213

Table 19: Runtimes of algorithms and environments and respective platform they were excecuted on. This results in a total CPU runtime of 7982.06h and GPU runtime of 20913.91 h (including 40.54h GPU hours for the runtime experiments).

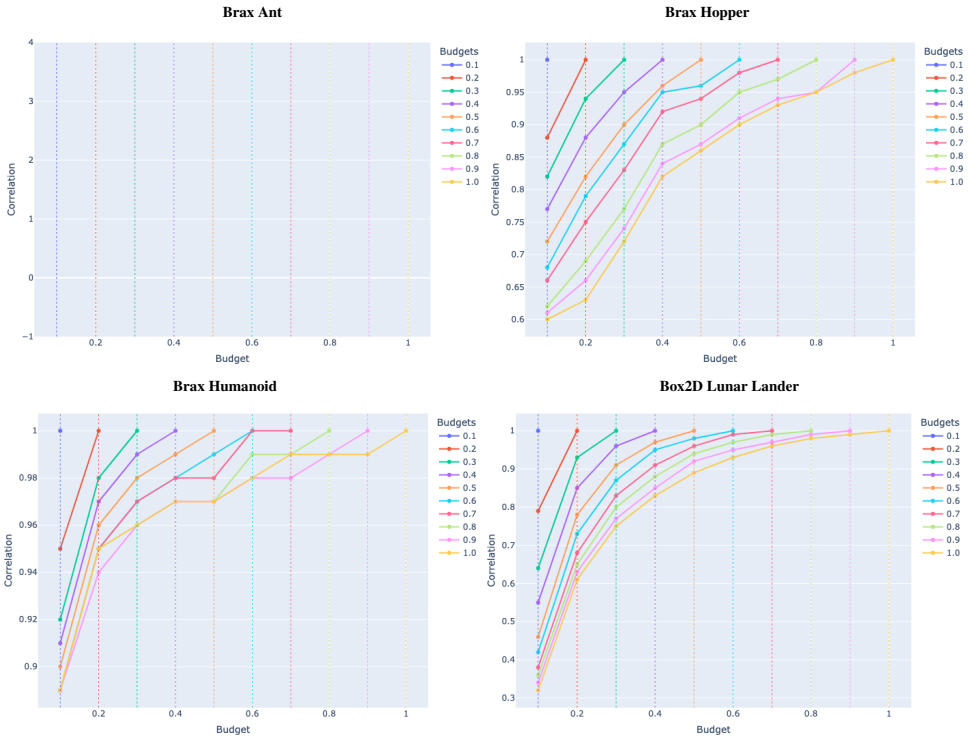

Figure 24: Budget Correlations for PPO: Brax and Box2D.

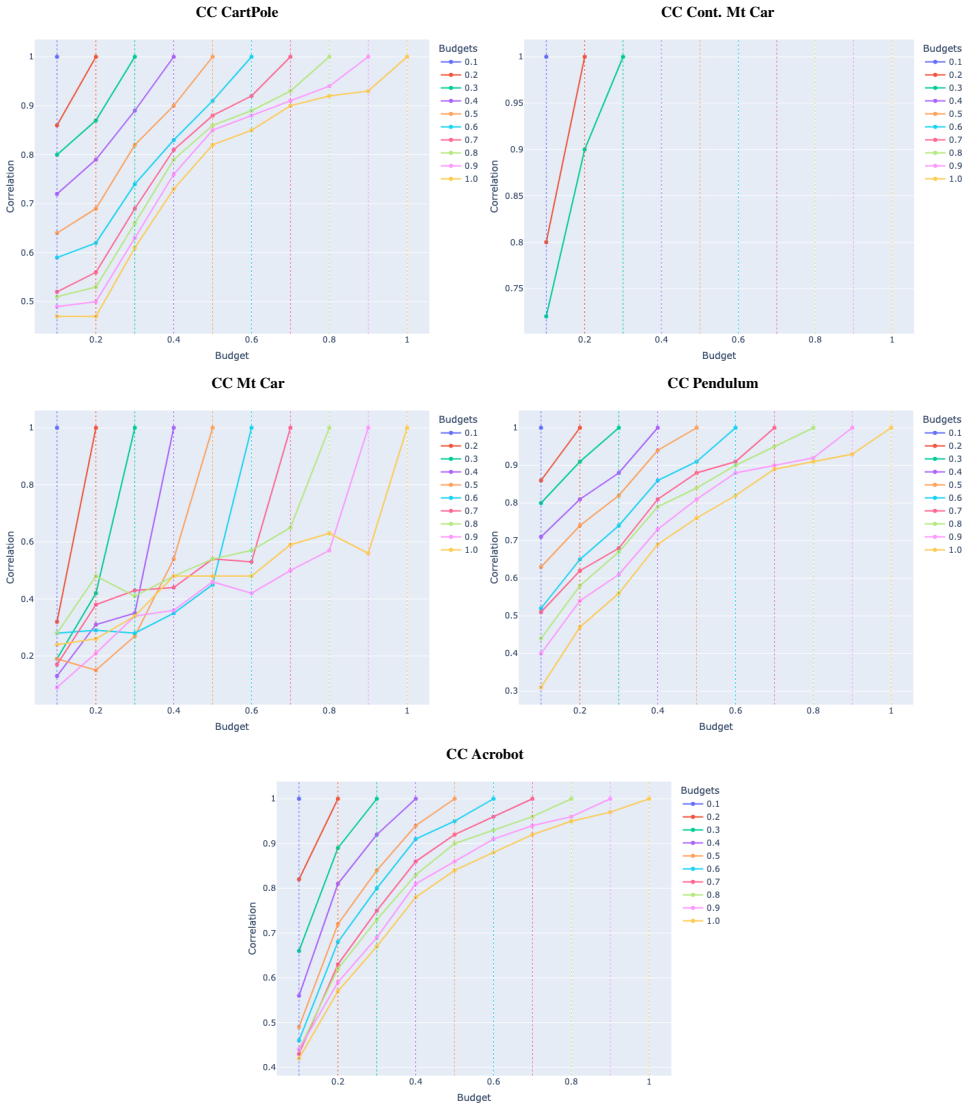

Figure 25: Budget Correlations for PPO: Classic Control.

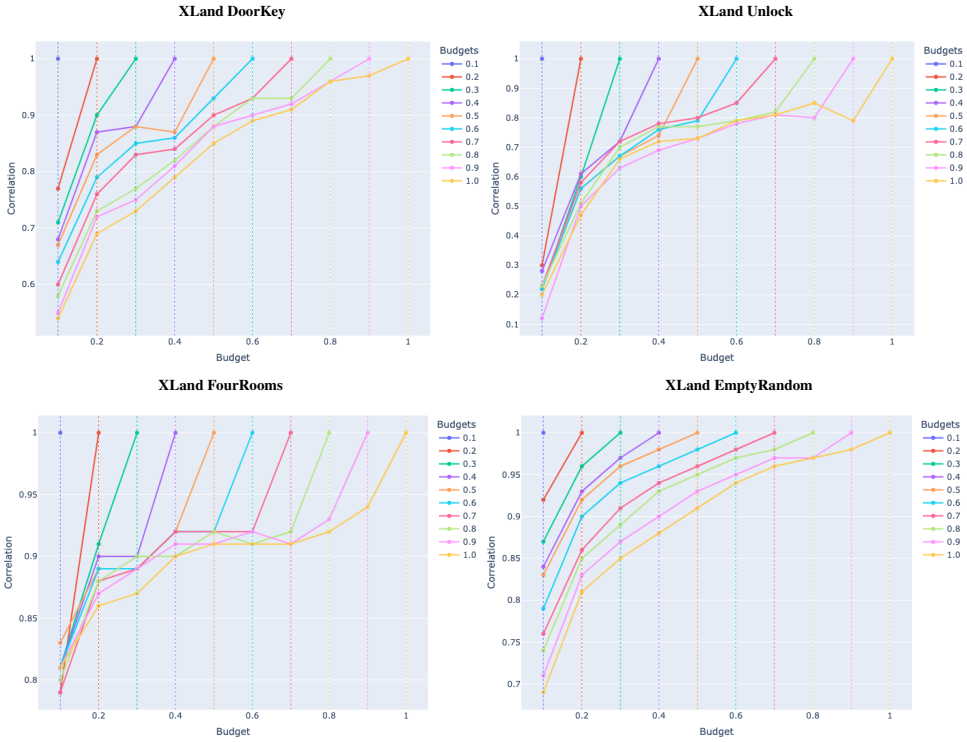

Figure 26: Budget Correlations for PPO: XLand.

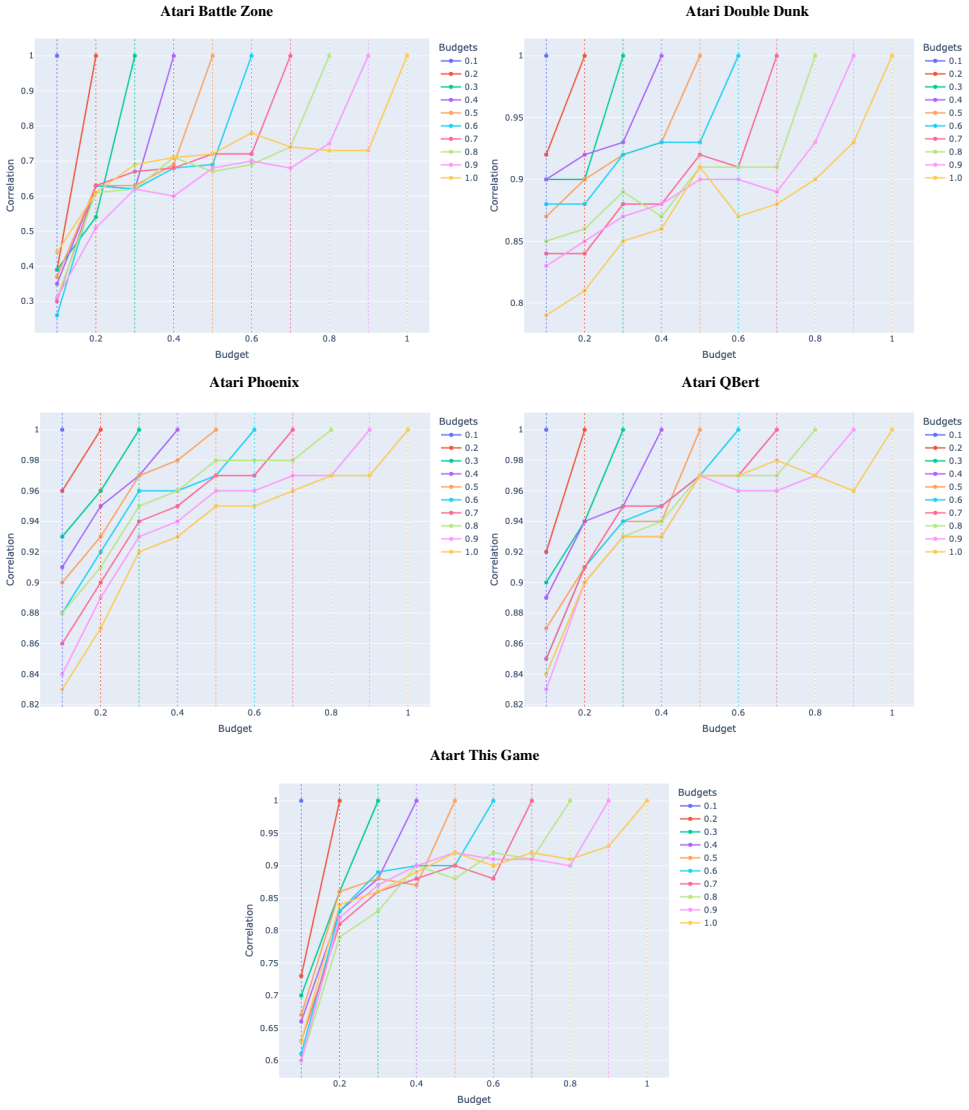

Figure 27: Budget Correlations for DQN: Atari.

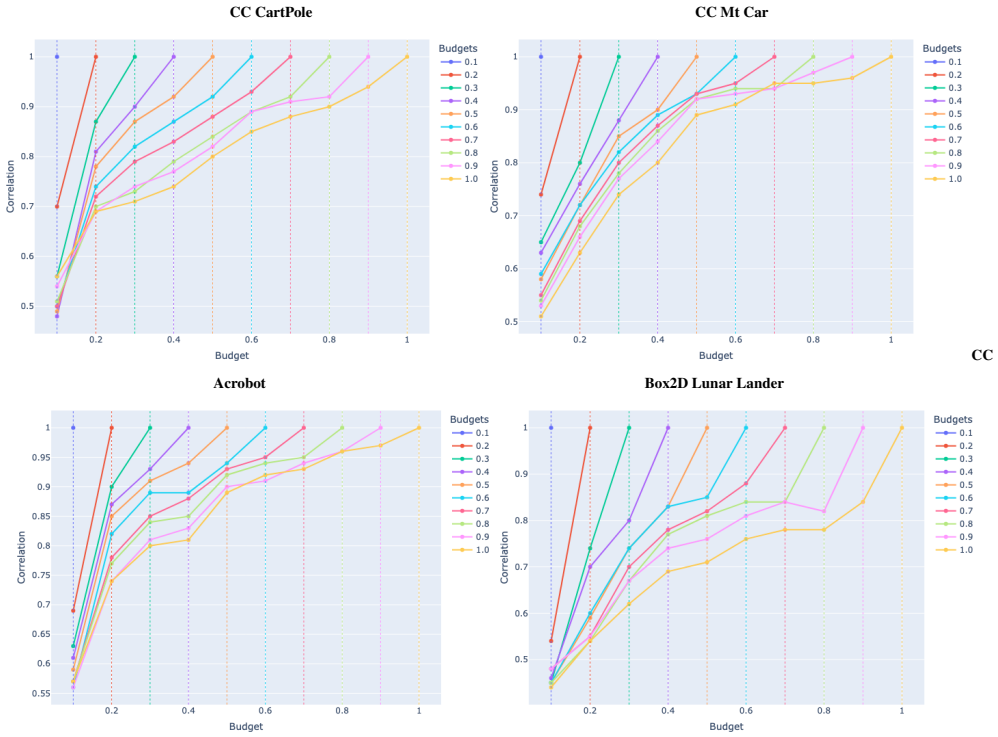

Figure 28: Budget Correlations for DQN: Classic Control and Box2D.

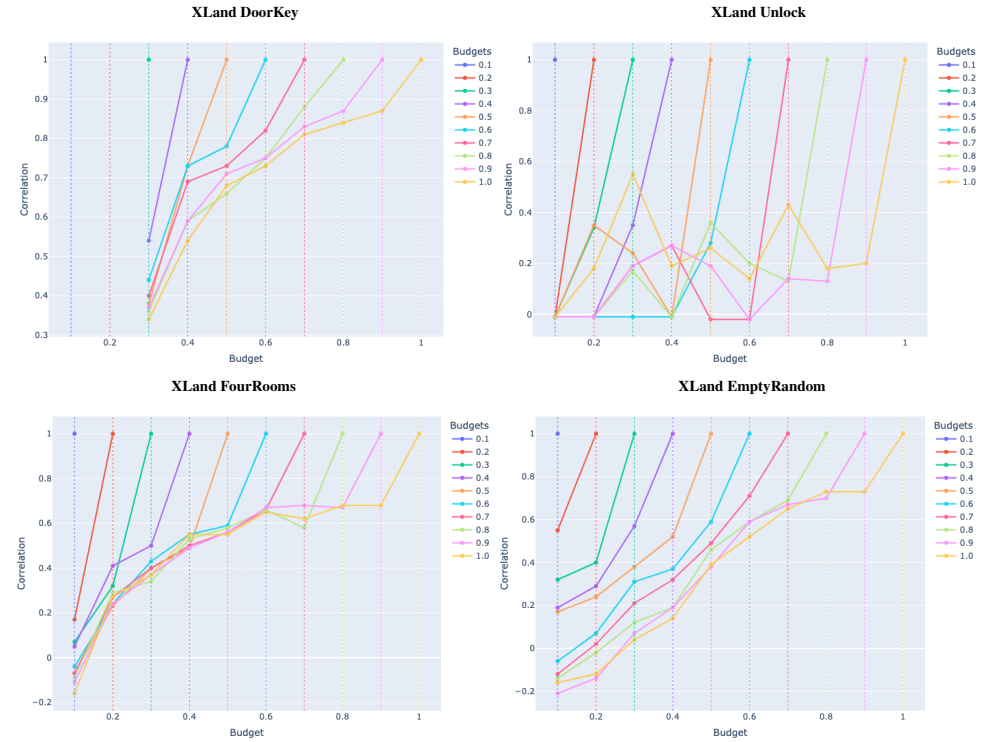

Figure 29: Budget Correlations for DQN: XLand.

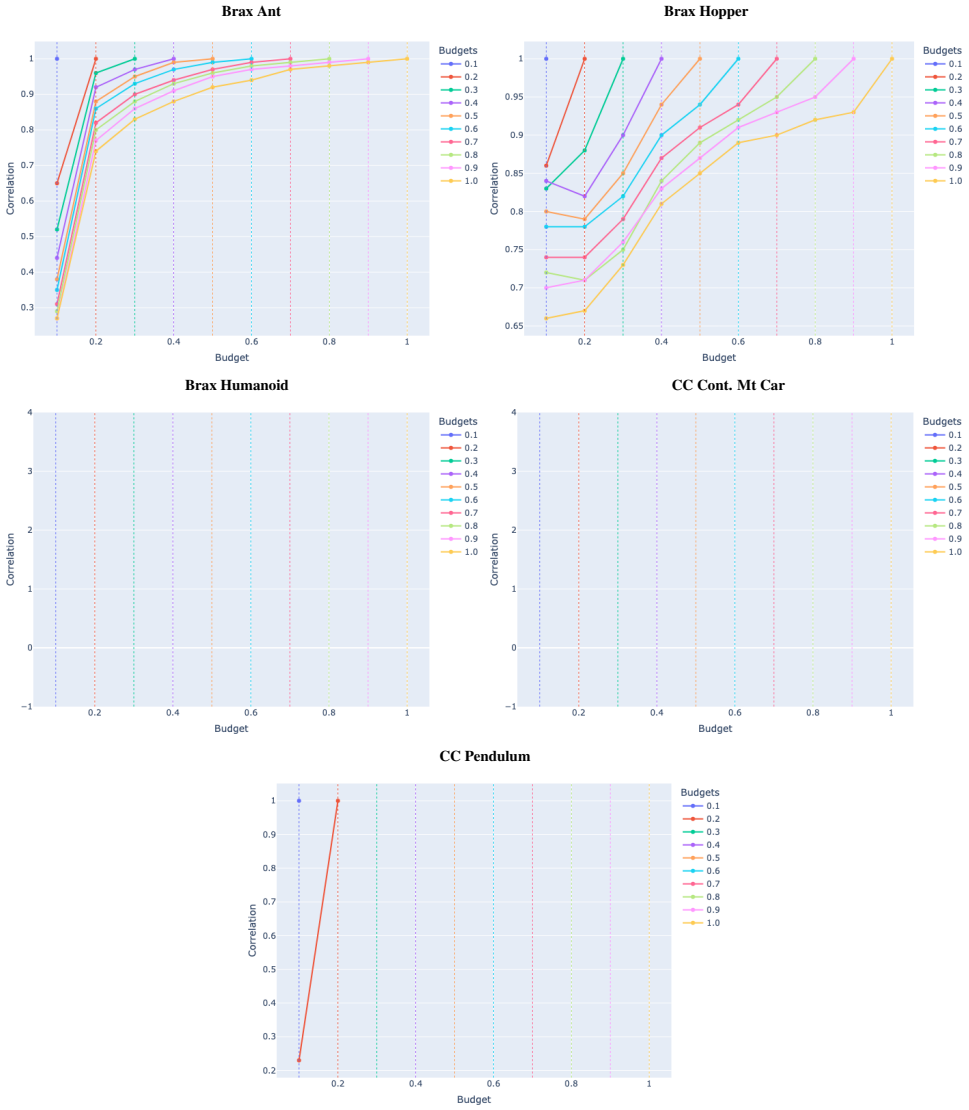

Figure 30: Budget Correlations for SAC.