Performance Prediction In Reinforcement Learning: The Bad And The Ugly

Julian Dierkes
Chair for AI Methodology
RWTH Aachen University
dierkes@aim.rwth-aachen.de

Theresa Eimer
Institut für Künstliche Intelligenz LUHAI
Leibniz University Hannover
t.eimer@ai.uni-hannover.de

Marius Lindauer
Institut für Künstliche Intelligenz LUHAI
Leibniz University Hannover
m.lindauer@ai.uni-hannover.de

Holger Hoos
Chair for AI Methodology
RWTH Aachen University, LIACS
Universiteit Leiden
hh@aim.rwth-aachen.de

Abstract

Reinforcement learning (RL) methods are known to be highly sensitive to their hyperparameter settings and costly to evaluate. In light of this, surrogate models that predict the performance of a given algorithm given a hyperparameter configuration seem an attractive solution for understanding and optimising computationally expensive tasks. In this work, we are studying such surrogates for RL and find that RL methods present a significant challenge to current performance prediction approaches. Specifically, RL landscapes appear to be rugged and noisy, which is reflected in the poor performance of surrogate models. Even if surrogate models are only used for gaining insights into the hyperparameter landscapes and not as replacements for algorithm evaluations in benchmarking, we find that they deviate from the ground truth significantly. Our evaluation highlights the limits of surrogate modelling for RL and cautions against blindly trusting surrogate-based methods for this domain. This calls for more sophisticated solutions for effectively using surrogate models in sequential model-based optimisation of RL hyperparameters.

1 Introduction

Surrogate models are used in different ways in automated machine learning (AutoML) research and practice: they can significantly reduce the cost of benchmarking, especially in domains with high computational cost [Eggensperger et al., 2021, Zela et al., 2022], and help with understanding the optimisation landscape of a given target algorithm via analysis methods such as fANOVA [Hutter et al., 2014a]. A domain that could significantly benefit from the cost reduction as well as the insights gained from using surrogates is reinforcement learning (RL), where hyperparameter optimisation is not yet well studied, partly due to the high cost of evaluation.

Recent work in RL re-emphasised how irregular the behaviour of RL algorithms can seem. Even on a well-tested benchmark such as the Atari learning environment (ALE; Bellemare et al. [2013], Machado et al. [2018]), it is surprising which algorithm is best for a given Atari environment [Farebrother and Castro, 2024]. Furthermore, hyperparameter configurations behave inconsistently across environments, with no clear pattern emerging as of yet [Ceron et al., 2024]. Given the immense computational cost of executing repeated runs of a benchmark like the ALE, the use of surrogates could significantly accelerate the development of algorithms as well as model selection and hyperpa-

18th European Workshop on Reinforcement Learning (EWRL 2025).

rameter optimisation (HPO) methods, and also facilitate deeper analysis of why we see such irregular behaviour in RL algorithms.

The challenge with the computational costs of RL and its notorious difficulty of tuning hyperparameters is that it likely makes performance predictions difficult. Indeed, we find that fitting surrogate models on the comparatively large ARLBench [Becktepe et al., 2024] dataset containing several RL tasks results in rather poor predictions. In this work, we investigate possible reasons for this phenomenon, attempt to boost surrogate performance and assess how much surrogate models can contribute to hyperparameter interpretability. Contextualising our results with data from supervised learning tasks from the LCBench [Zimmer et al., 2021] and PD1 [Wang et al., 2024] benchmarks paints the picture of a comparatively complex and noisy hyperparameter landscape far from what Pushak and Hoos [2018] found to be benign landscapes for traditional machine learning methods. Our evaluations suggest that significant amounts of data are needed to fit reliable RL surrogates; even if we are merely interested in hyperparameter importance (LPI; Hutter et al. [2014a], Biedenkapp et al. [2018]) or ablation paths Fawcett and Hoos [2016], surrogates are not consistently showing the same qualitative behaviour as the algorithms whose performance they model. Therefore, we recommend not relying on surrogates when analysing hyperparameters in RL, but to use methods that do not involve performance prediction. At the same time, we believe RL can be a challenging domain of considerable interest in the context of improving surrogate modelling methods.

In summary, our contributions are: i. an investigation of the difficulty of the hyperparameter landscapes of RL, particularly in terms of unimodality and the influence of randomness; ii. a thorough evaluation of surrogate model performance on RL tasks, including scaling with the number of configurations and runs; iii. a comparison of surrogate-based hyperparameter interpretability methods (LPI, ablation paths) with their analytical ground truth.

2 Related Work

Before discussing our investigation, we will give an overview of the context of our work within AutoRL, the construction of surrogate benchmarks, and surrogates for interpreting HPO landscapes.

AutoRL. Automated reinforcement learning (AutoRL; Parker-Holder et al. [2022]) is concerned with automatically adapting RL algorithms to any given setting. This includes HPO [Parker-Holder et al., 2020] just as much as neural architecture search [Miao et al., 2022], task scheduling [Portelas et al., 2020, Jiang et al., 2021] and more. Notably, several previous studies have found unexpected RL algorithm behaviour concerning hyperparameters, e.g. related to noise between different runs [Eimer et al., 2023] or significant differences between algorithms [Farebrother and Castro, 2024]. To further explore this and to enable principled comparisons of HPO approaches for RL, the tabular benchmark HPO-RL-Bench [Shala et al., 2024] and ARLBench have been proposed [Becktepe et al., 2024]. We focus on the latter, since it provides a dataset of random search data, instead of the grid data of HPO-RL-Bench, across a wider variety of environment domains. Furthermore, the search space underlying HPO-RL-Bench is heavily limited, focusing on only 2–3 hyperparameters, with an additional 2 for the architecture used in some algorithms. As our goal is to investigate the optimisation landscape of RL, this constraint would severely limit the generality of our insights.

Surrogate-Based Benchmarking. Since the study of HPO is inherently costly, due to the need for many repeated function evaluations, the community started publishing datasets containing evaluation results as a fast and efficient way for benchmarking: so-called tabular benchmarks. Notable examples for machine learning tasks include NASbench 101 [Ying et al., 2019] and 201 [Dong and Yang, 2020], HPO-B [Pineda et al., 2021], LCBench [Zimmer et al., 2021] and the recent PD1 [Wang et al., 2024]. Since these datasets are limited to a grid-based search space, however, their utility remains limited. Thus, surrogate models capable of predicting configurations not in the dataset have been proposed [Hutter et al., 2014b, Eggensperger et al., 2015, 2018a]. Surrogate-based benchmarks are now important measures of HPO performance, e.g. as part of HPOBench [Eggensperger et al., 2021], in NASBench 301 [Zela et al., 2022] or YAHPOGym [Pfisterer et al., 2022]. Surrogate modelling techniques are also continuously improved upon through ideas such as the direct modelling of the distribution of performance over multiple independent runs [Eggensperger et al., 2018b, Li et al., 2024]. While the dominant mode of a surrogate model used in benchmarking predicts a single performance value, there are also approaches for predicting full learning curves from either a configuration or incomplete runs [Klein et al., 2017, Adriaensen et al., 2024].

Surrogates for HPO Interpretability. Surrogate models are also used for gaining insights into the behaviour of target algorithms; they can assist by simulating the evaluation of additional configurations for approaches such as ablation paths [Biedenkapp et al., 2017], local hyperparameter importance [Biedenkapp et al., 2018], HyperSHAP [Wever et al., 2025] or partial dependency plots [Moosbauer et al., 2021]. Some specific surrogate model classes, such as random forests, can even be directly used to compute insights into hyperparameter importance [Hutter et al., 2014a]. Systems such as DeepCave [Sass et al., 2022] provide several such tools packaged together, such that even HPO runs with a constrained budget can easily provide meaningful insights into the response of a given target algorithm to hyperparameter settings.

3 Datasets

To analyse and compare the performance of surrogate models, we used four tabular benchmarks. Each benchmark maps hyperparameter configurations of an RL or supervised learning scenario to the corresponding training performance.

ARLBench: *ARLBench* [Becktepe et al., 2024] is a hyperparameter optimisation benchmark that includes performance landscapes for three RL algorithms: PPO [Schulman et al., 2017], DQN [Mnih et al., 2013] and SAC [Haarnoja et al., 2018]. These landscapes cover 21, 13 and 8 different environments and contain 9, 12 and 11 hyperparameters, respectively. Each landscape consists of 256 configurations, with 10 independent runs per configuration to account for performance variability.

Lunar Lander 2048: To investigate how surrogate model performance scales with dataset size, we extend the performance landscapes for the Lunar Lander environment to 2048 configurations and 30 independent runs per configuration for all three RL algorithms using the benchmark implementation provided by ARLBench. We refer to this dataset as *Lunar Lander 2048*.

LCBench: *LCBench* [Zimmer et al., 2021] provides performance landscapes for hyperparameter tuning of simple multi-layer perceptrons on tabular datasets. It comprises 35 OpenML tasks [Vanschoren et al., 2014], each with 2000 configurations evaluated using 3 independend runs per configuration and tuned with 7 different hyperparameters. For comparability with ARLBench, we randomly subsampled 256 configurations from each task, referring to the resulting benchmark as *LCBench 256*, while retaining the original dataset under the name *LCBench*.

PD1: *PD1* [Wang et al., 2024] is a performance dataset of CIFAR-100 [Krizhevsky, 2009] training runs. It includes four tasks, each with 500 configurations of 5 different hyperparameters evaluated using a single run. Again, we randomly subsample 256 configurations for comparability to *ARLBench*.

For all datasets, we normalise the performance values between 0 and 1, using appropriately chosen bounds. Further details about the datasets can be found in the Appendix A.

4 Fitting Surrogate Benchmarks For RL

To train surrogate benchmarks based on the ARLBench data, we first selected commonly used surrogate model classes to tune for performance on each ARLBench task. We did so for predicting either the final performance, maximum performance or area under the performance curve of each configuration over the 10 seeds in the ARLBench dataset. We performed this initial comparison on random forests (RFs, Breiman [2001]), Gaussian processes (GPs, Rasmussen and Williams [2006]), support vector machines (SVMs, Cortes and Vapnik [1995]) and XGBoost (XGB, Chen and Guestrin [2016]). For RFs, GPs and SVMs, we used the implementation of Scikit-learn [Pedregosa et al., 2011]. All models have been tuned using random search with 200 trials, except GPs, which have been tuned for only 50 runs, since they have substantially fewer hyperparameters to tune. The performance of a model was evaluated using 10-fold cross-validation. Each model was fitted 5 times to account for the variability in surrogate training. The search spaces of the surrogate models are similar to NAS-Bench-301 [Zela et al., 2022] and described in the Appendix B. We compared the performance of our surrogate models on LCBench to YAHPOGym to ensure the correctness of our implementation.

Following Bansal et al. [2022], we evaluated performance using Kendall's Tau to assess the consistency of the surrogate prediction rankings, and RMSE, to measure the absolute accuracy of the predictions. Given its prominence in the literature, we focused our analysis on predicting the final performance of configurations unless otherwise mentioned. A more detailed comparison of the

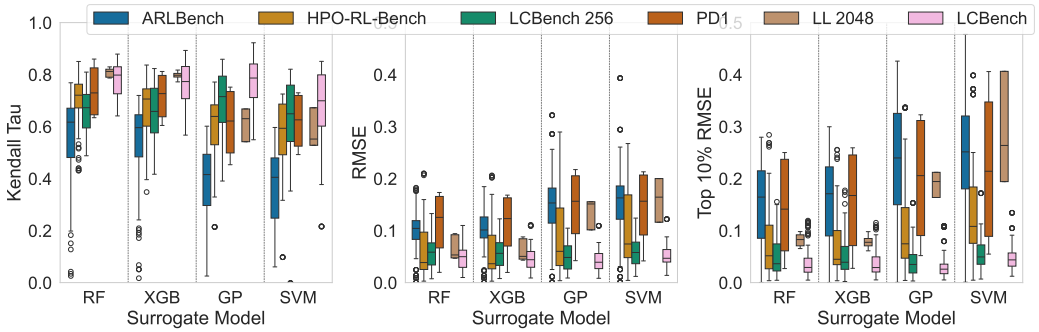

Figure 1: Prediction performance of RFs, XGB, GPs and Support SVM. The top 10 RMSE is the RMSE computed on only the 10 best-performing configs in the landscape. The landscapes of ARLBench and LunarLander 2048 consist of three seeds per config to be comparable to LCBench.

performance metrics can be found in Appendix B. We observed a few suboptimal trends in the surrogate performance values in Figure 1. Firstly, there are several outliers with very low Kendall Tau and very high RMSE (up to 0.2), indicating tasks with poor surrogate fits. The median RMSE of the top 10% of configurations is only just below 0.2 for the best surrogate model, indicating that even for very good configurations, predictions are far from precise.

When comparing these results to the same surrogate fitting procedure for tasks from supervised deep learning using the data from LCBench [Zimmer et al., 2021] and PD1 [Wang et al., 2024], we found similar effects, albeit with less drastic outliers for PD1 and lower errors for LCBench. Between these two benchmarks, PD1 contains the more difficult and computationally expensive tasks. Therefore, our results let us draw a difficulty curve from the moderate errors on LCBench to worse RMSE on PD1 with fairly good Kendall Tau and no outliers, up to worse Kendall Tau and large outliers on ARLBench. Therefore, the question becomes: what makes ARLBench performance hard to predict and how can additional data improve our surrogate models?

4.1 Understanding HPO Landscapes of RL

An easy-to-model performance landscape is relatively smooth and unimodal, as these features allow for accurate predictions on neighbouring configurations. The more we move away from these characteristics, the more effort we likely need to expend for data collection and model training. Therefore, we characterised the ARLBench landscapes by its performance distribution and unimodality to gauge how well suited it is for performance prediction.

Performance Distribution. ARLBench, LCBench and PD1 consist of several different benchmark tasks of various difficulty levels. Looking at the distribution of performance values in Figure 2, we observe that for LCBench, several tasks show narrow performance spread, meaning the worst and best performances are close together. Tasks that utilise less than 50% of the full performance spectrum make up 49% of the LCBench, 25% of the PD1 and 15 % of the ARLBench tasks we considered. Other tasks show fairly large spread, covering over 80% of the performance spectrum. This is the case for 17% LCBench tasks and 62% ARLBench tasks, but not for any PD1 task. The width of the performance spread does not necessarily imply anything about the difficulty of the landscape, but it is noticeable that there seems to be a greater variety in the observed performance values on ARLBench than on LCBench and PD1 overall. Furthermore, more ARLBench tasks than LCBench tasks have a "bottom-heavy" performance distribution, where most of the performance values are clustered towards the bottom of the spectrum, as can be seen in Figure 2. This could imply narrower optima in the landscape for ARLBench than in that for LCBench.

Unimodality. To obtain deeper insights into the shape of the ARLBench landscapes, we inspected its unimodality similarly to Pushak and Hoos [2018]. We measured what percentage of points can be reached from the optimum while performance either stays the same or decreases. This reachability ratio thus quantifies the extent to which we can follow the trajectories in the landscape without ending in local optima. Since the confidence intervals of some tasks, particularly from ARLBench, are very large, we plotted the reachability ratio for standardised performance intervals around each point (see

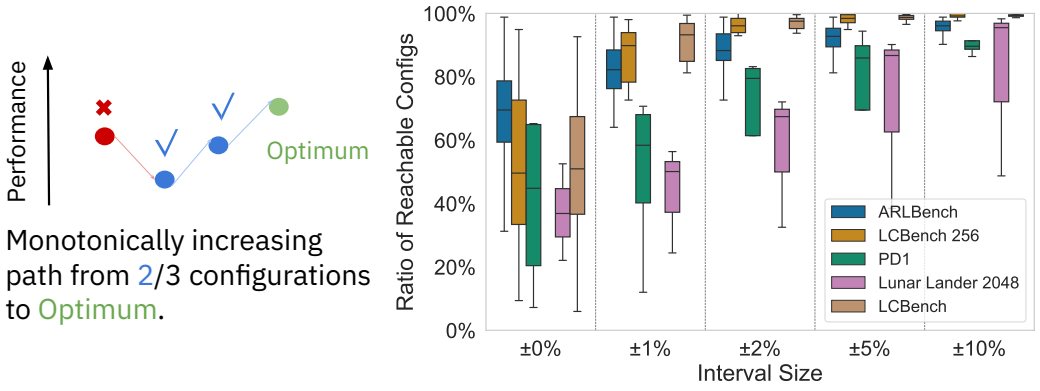

Figure 3: **Left:** One-dimensional illustration of reachability in our analysis. **Right:** The plot shows the percentage of reachable configurations via monotonically decreasing paths from the best-performing point within performance intervals. For intervals > 0, supervised learning landscapes show significantly greater unimodality, which is an unexpected result, considering the strong unimodality observed in AutoML landscapes by Pushak and Hoos [2022].

Figure 3). To ensure that the performance intervals are comparable between landscapes with different spreads, as observed in Figure 2, we normalised the performances in each landscape by the respective minimal and maximal performance values.

We observed that with a performance interval of 0, that is, traversing only the mean performances, ARLBench tasks show a median reachability ratio of around 70%, while both LCBench variations are considerably worse, at less than 60% and 40%. This trend is reversed once we add performance intervals. With an interval of only ± 0.01 around each point, the reachability ratio for both LCBench variations improves to around 90% while for ARLBench, it only climbs to just above 80%. At a performance interval of ± 0.1, LCBench is roughly unimodal, while no ARLBench task reaches a reachability ratio of 1. We interpret this result to indicate that while LCBench has a more rugged optimisation landscape, the ruggedness is caused by smaller variations that are largely smoothed out within a ± 0.1 performance interval. The ARLBench landscapes, on the other hand, seem less rugged in principle, but show larger features that are not smoothed out as easily. From an optimisation perspective, such landscape features can be harder to navigate than smaller irregularities in LCBench.

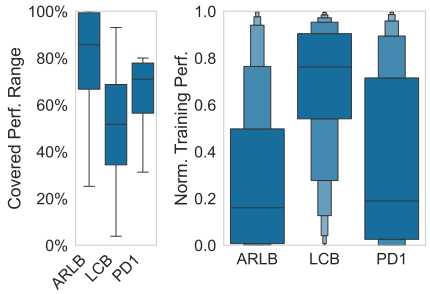

Figure 2: **Left:** Boxplots of performance range coverage relative to the theoretical minimum and maximum achievable on the task. **Right:** Boxplots of performance across configurations. In comparison, ARLBench shows a heavier concentration of low-performing configurations.

These results imply that, on average, the RL tasks in ARLBench show greater performance spreads, narrower optima and a harder-to-navigate landscape than the supervised learning tasks in LCBench. Combined, these effects let us hypothesise that the amount of ALRBench performance data needed for performance prediction of the same quality as for LCBench is higher than we currently available to us, potentially by a significant margin. Empirically, we confirmed this effect for LunarLander (see Figure 4), one of the most stable benchmarks with average performance scores. More configurations lead to a higher Kendall Tau and more stable predictions, though the overall improvement from 256 configurations to 2048 configurations remains limited. Very good scores would likely require a much greater increase in the number of available configurations, more targeted data in poorly performing regions or other remedies, such as ensembles of surrogate models.

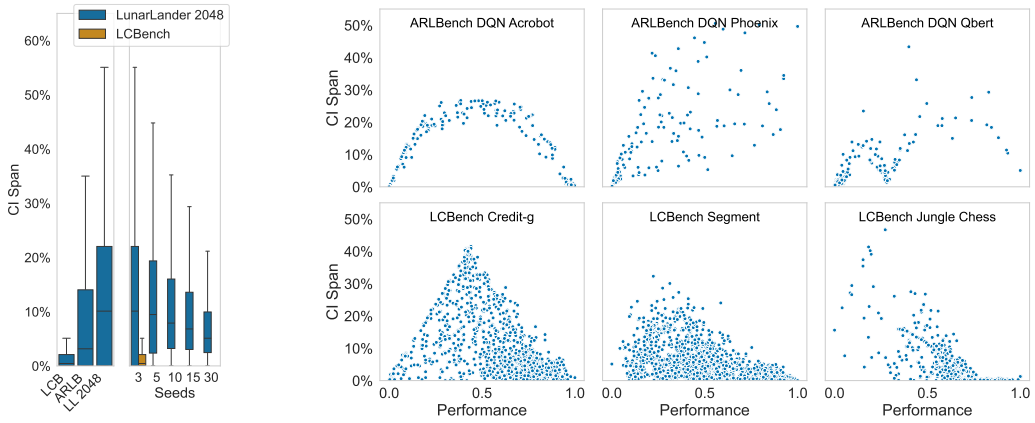

Figure 5: **Left:** Spread of 95% CIs across datasets and for increasing number of seeds in ARLBench. **Right:** Three distinct patterns between configuration performance and CI size (top for ARLBench, bottom for LCBench). Unlike ARLBench, LCBench consistently exhibits very small CIs at high performance levels. The best fit of a polynomial of degree two is shown in red.

4.2 The Influence of Randomness

Looking at the size of confidence intervals (CIs) per configuration between ARLBench and LCBench (see Figure 5), we noticed a trend towards much larger CIs in ARLBench and high deviations in CI size. While LCBench shows CIs with an upper whisker up to around 0.051 with a median of 0.004 for three seeds, the CIs for ARLBench extend up to 0.57 and have a much larger median with 0.1. This makes predicting the performance of a configuration quite hard on ARLBench, as the variations between two runs of the same configuration can be significant. It also makes modelling performance harder, since we expect to require a larger number of runs per configuration for an accurate prediction.

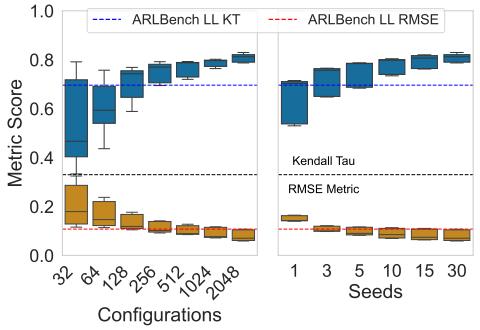

Figure 4: **Left:** Performance gains of LunarLander surrogate models as the number of configurations increases (with 30 seeds per configuration). **Right:** Performance gains as the number of seeds per configuration increases (with 2048 configurations).

If we could model the variations between runs, however, we might be able to considerably improve predictions. Therefore, we tested whether the variation between runs is heteroscedastic, i.e. whether the evaluation noise is inconsistent across configurations. We assessed heteroscedasticity using Levene's test, which evaluates the null hypothesis that the variances across groups are equal (i.e., the data is homoscedastic). The test was first applied across all configurations within each environment. To mitigate potential bias from extreme values, we repeated the test, considering only those configurations whose performance values did not coincide with the minimum or maximum observed performance in the respective environment. A significance level of 0.05 was used to determine statistical significance. Homoscedasticity is rejected in nearly all cases across both testing setups, with p-values generally below $1 \cdot 10^{-8}$ and often much smaller, indicating strong evidence of unequal variance. The only exception occurs in the Mountain Car environment with PPO, under the reduced configuration set, where the test fails to reject homoscedasticity. The likely reason for this is that only two of the 256 configurations do not lie on the performance boundary, and thus, the test was performed on only these two configurations; when including all configurations, homoscedasticity is rejected in this scenario as well. This result shows that we cannot expect to model the variations between seeds even with surrogate models that can take randomness into account, such as BNNs [Goan and Fookes, 2020].

When looking at the distribution of CI size per ARLBench environment concerning performance shown in Figure 5, we found that ARLBench also shows a more diverse and complex behaviour of the CIs of configurations with increasing performance values compared to LCBench. Some environments show a smooth progression of low CIs to high CIs with medium performance and back to low CIs for very well-performing configurations again. This, unfortunately, is not always the case; several environments seem to show higher-dimensional relationships between CI size and performance, and there are even cases with a linear increase in CI as performance improves. Particularly the latter case presents a challenge for training surrogates, as the already large set of ALRBench data seems to be far from sufficient for good predictions in high-performance regions of the optimisation landscape.

We ablated the number of seeds used to fit surrogates for LunarLander in Figure 4. This is an environment with lower CIs for both good and bad configurations, i.e. a comparatively well-behaved task, where we would expect better predictions with more seeds. Again, we observed a steady but decreasing improvement, leading to a fairly small difference between the 10 seeds of the original ARLBench data and our maximum of 30. As with the number of configurations, the data requirements for well-performing surrogates can be assumed to be much larger than our tripling of seeds.

4.3 Are RL Surrogate Benchmarks Feasible?

Our results show several trends that can explain the relatively poor performance of our ARLBench surrogates: the ARLBench optimisation landscape is much more rugged and less uniform than the benign optimisation landscapes of smaller-scale supervised machine learning scenarios [Pushak and Hoos, 2018] or simple supervised deep learning tasks. PD1 shows similar non-uniformity metrics, indicating that advanced deep learning tasks generally have more complex optimisation landscapes. Additionally, ARLBench shows large amounts of variation for each configuration, which cannot easily be modelled. This makes prediction very difficult, especially in cases where the CI size increases with configuration performance. Empirically, we have seen that more configurations and more seeds yield improved surrogate performance for the average ARLBench task of LunarLander in Figure 4, but our results clearly indicate diminishing returns.

These findings suggest that current surrogate modelling methods are unlikely to yield reliable surrogate benchmarks for RL, given the high cost of data collection. This does not mean, however, that surrogate benchmarks are impossible. A key element will likely be finding a way to model randomness in the landscape, and while this may not easily be possible in the black-box prediction setting we considered here, it is conceivable that future methods use information about the environment and algorithm trajectory for this purpose. Thus, RL can function as a challenging benchmark for new surrogate modelling methods.

5 Using Surrogates To Interpret RL HPO Landscapes

Beyond surrogate benchmarking, an important use of performance prediction in AutoML is the generation of insights into the HPO landscape. Examples are surrogate-based hyperparameter importance (e.g. fANOVA; Hutter et al. [2014a]) or using surrogates for the cheap evaluation of metrics that would require additional configuration evaluations (e.g. in the context of ablation analysis; Fawcett and Hoos [2016]). Even if accurate surrogate benchmarks for RL tasks appear to be out of reach for now, RL surrogates could still perform well enough to gather such useful insights into HPO characteristics of these scenarios. To evaluate the quality of surrogate-based insights in RL, we chose methods that allow us to either evaluate new configurations or substitute a surrogate model: LPI [Biedenkapp et al., 2018] and ablation paths [Fawcett and Hoos, 2016]. We focus once again on our example case of Lunar Lander, since here, we can explore how the quality of the insight improves with more surrogate training data. Given their high performance, we focus our analysis on the RF surrogate models in the remainder of this paper.

5.1 Local Hyperparameter Importance

LPI quantifies the importance of each hyperparameter within a given configuration by measuring the proportion of performance variance explained when that hyperparameter is varied. This provides a configuration-specific assessment of which hyperparameters are most influential for a particular setup. A more detailed explanation is provided in Appendix C.

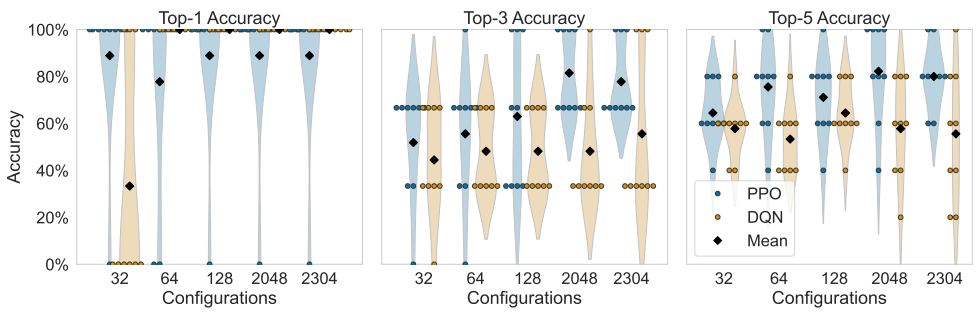

Figure 6: Accuracy of LPI predictions by surrogate models for PPO and DQN on the LunarLander 2048 benchmark. We report Top-1, Top-3 and Top-5 accuracy, averaged over three configurations (best, 25th percentile and median), each evaluated with 30 seeds.

To evaluate the accuracy of surrogate models in estimating LPI, we computed ground-truth LPI values for three representative configurations in the *LunarLander 2048* benchmark for PPO and DQN: the best-performing, 25th percentile and median configuration. Each configuration was evaluated with 30 independent seeds. We then compared these ground-truth LPIs to the corresponding estimates produced by surrogate models trained on subsets of the LunarLander 2048 dataset, aiming to assess the ability of the model to recover configuration-specific hyperparameter importance. The raw LPI results and further experimental details can be found in Appendix C.

Figure 6 shows the accuracy of surrogate-based LPI predictions. We report three metrics: top-1 accuracy (correctly identifying the single most important hyperparameter), top-3 accuracy (fraction of correctly predicted hyperparameters among the top three) and top-5 accuracy. We observe that the average top-1 accuracy is between 60% and 90% for all models, indicating that the surrogate models correctly identify the most important hyperparameter in many cases. Top-3 and top-5 accuracies are similar, ranging between 60% and 80%, which suggests that, although the exact rankings are sometimes inaccurate, the models can capture broader trends in hyperparameter importance.

However, we do observe that the surrogate models are not necessarily able to reliably predict importances for the full set of hyperparameters, particularly less important ones.On average, PPO and DQN have 5.2 and 8.6 hyperparameters with an importance above 5%, while the surrogate models estimate only around 3.6 and 3.5, respectively. Therefore, the surrogate models cannot be relied upon for a full picture of hyperparameter importance, and their output should be seen as a rough guidance rather than an accurate trend. This is compounded by the fact that the absolute difference between ground truth and importance estimate can be quite large even if the most important hyperparameter is correctly identified (e.g. up to a factor of 2 in each of the PPO configurations we consider). Surrogate-based LPI can therefore be used to extract insights such as the three most important hyperparameters for a task, but not to make judgments on absolute importance or differences and rankings between important hyperparameters. Thus, their usefulness is severely limited compared to the ground truth.

Interestingly, increasing the number of configurations used for training the surrogate models does not substantially improve top-1 or top-5 accuracy. This suggests that accurately modelling local hyperparameter importance remains challenging, even for surrogate models trained on large datasets.

5.2 Ablation Paths

Ablation paths quantify both the importance and interdependencies of hyperparameters by ordering changes from a source configuration to a target configuration based on the performance improvement induced by each change. Predicting ablation paths is a challenging task due to the complexity of the ARLBench landscapes and the high errors observed in our surrogate models. A more detailed explanation is provided in Appendix D. To assess surrogate quality for ablation paths, we selected source configurations for LunarLander 2048 PPO and DQN across several performance percentiles (2nd, 13th, 25th, 38th, 50th, 63rd, 75th and 99th), using the overall best configuration as the target. We then computed the ground-truth ablation paths with 30 seeds per configuration and compared them to those predicted by our tuned surrogate models.

Raw results are shown in Appendix D. Ablation paths starting from the 2nd percentile exhibit negligible performance changes compared to the target configuration and were excluded from further analysis. To quantify surrogate performance, we focused on cases where the ground-truth path contained performance shifts greater than 10% of the performance spread of a task and evaluated whether the surrogate identified the correct hyperparameter responsible. For these cases, we computed top-1, top-3 and top-5 accuracy metrics.Top-1 accuracy, where the surrogate identifies the exact hyperparameter responsible for the shift, is generally around 20%, except for the DQN surrogate trained with only 32 configurations, where it rises to 40$. Top-3 accuracy ranges from 50% to 80%, and top-5 accuracy is approximately 95% for PPO and 80% for DQN.

Accurate identification of hyperparameter order, especially of hyperparameters with the largest impact, is crucial for ablation paths. An incorrect hyperparameter at any step can significantly alter the remainder of the path. Low top-1 accuracy is therefore a serious limitation. Additionally, increasing the number of configurations used for training does not lead to consistent improvements and sometimes results in worse performance, which is counterintuitive and further suggests that the surrogate models are not able to fully reflect landscape properties important for ablation paths.

We also examined the difference in predicted performance between the top-ranked hyperparameter of a surrogate and the correct one (see Appendix D). These differences were often large, indicating that the surrogate models do not simply make minor errors due to noise, but fully misattribute performance increases. This conclusion is supported by visual inspection of the ablation paths in Appendix D, where neither the sequence of changes nor the overall shape of the surrogate paths resembles the ground truth well. Considering the Top-10 RMSE values in Figure 1 and the result in Figure 2 showing that RL landscapes contain a large proportion of low-performing configurations, we hypothesise that surrogate models particularly struggle in accurately modelling high-performing regions. These regions, however, are the most important for ablation path prediction.

In summary, the surrogate models we trained are not sufficiently accurate to support reliable ablation analysis on ARLBench tasks. Unlike in the case of LPI, where at least the broad trends hold, ablation paths based on our surrogate models do not reflect the ground truth and are unlikely to yield any new insights into the HPO landscape.

6 Ugly Landscapes, Bad Performance, Few Insights

Unfortunately, we have not been able to train surrogate models of RL algorithms that can be used for surrogate benchmarking or HPO landscape analysis. Indeed, our analysis shows that surrogate modelling may be very difficult for RL in general and possibly also for more complex deep learning problems. The optimisation landscapes we have seen are not unimodal, RL tasks induce irregular noise patterns we cannot model, surrogates have large prediction errors even with tuning and additional data collection, and surrogate-based HPO insights are unreliable even when trained on large amounts of performance data. In short, we were unable to use surrogate models for RL in any of the ways in which they have benefited the AutoML community thus far.

Nonetheless, we do not believe that performance prediction in RL is completely infeasible; however, the current paradigm appears unlikely to produce good results with realistic amounts of training data. It is possible that adapting approaches from learning curve prediction, which are based on partial evaluations for predicting final performance [Adriaensen et al., 2024], could mitigate this problem. Potentially deeper insights into the reasons for the extreme variations between RL runs may pave the way towards better modelling. In any case, we hope this clear failure case of established methodology will lead to new solutions for training surrogate models in the future.

For now, however, we have to warn against the use of performance prediction in use cases requiring precision. Even for environments with seemingly low error rates within the ARLBench tasks, such as Lunar Lander, surrogate predictions are often faulty and will not reliably match characteristics of the underlying performance landscape. HPO insights should be gathered with additional function evaluations instead of fully surrogate-based methods.

The question of how this influences HPO methods remains open. Some previous work has found that BO performs close to random search on RL environments [Becktepe et al., 2024], but the literature is inconsistent thus far [Shala et al., 2024]. On tasks comparable to the PD1 data, BO has also not performed well for learning rate optimisation, with significantly worse results than on simpler tasks

with a much more uniform landscape [Henheik et al., 2025]. Overall, however, the lack of empirical investigation into this area prevents us from linking decreasing landscape uniformity and increasing difficulties for surrogate-based optimisers. We believe further research is required to confirm whether the challenges for surrogate predictions we find here are a limiting factor for optimiser performance in these highly relevant areas of machine learning.

Acknowledgements

Theresa Eimer acknowledges funding by the German Research Foundation (DFG) under LI 2801/10-1.

References

S. Adriaensen, H. Rakotoarison, S. Müller, and F. Hutter. Efficient Bayesian learning curve extrapolation using prior-data fitted networks. In *Proc. of NeurIPS*, 2024.

A. Bansal, D. Stoll, M. Janowski, A. Zela, and F. Hutter. JAHS-bench-201: A foundation for research on joint architecture and hyperparameter search. In *Proc. of NeurIPS*, 2022.

J. Becktepe, J. Dierkes, C. Benjamins, A. Mohan, D. Salinas, R. Rajan, F. Hutter, H. Hoos, M. Lindauer, and T. Eimer. ARLBench: Flexible and efficient benchmarking for hyperparameter optimization in reinforcement learning. In *Seventeenth European Workshop on Reinforcement Learning*, 2024.

M. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47, 2013.

A. Biedenkapp, M. Lindauer, K. Eggensperger, C. Fawcett, H. Hoos, and F. Hutter. Efficient parameter importance analysis via ablation with surrogates. In *Proc. of AAAI*, 2017.

A. Biedenkapp, J. Marben, M. Lindauer, and F. Hutter. CAVE: Configuration assessment, visualization and evaluation. In *Proc. of LION*, 2018.

L. Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.

J. S. O. Ceron, J. G. M. Araújo, A. Courville, and P. S. Castro. On the consistency of hyper-parameter selection in value-based deep reinforcement learning. *Reinforcement Learning Journal*, 1, 2024.

T. Chen and C. Guestrin. XGBoost: A scalable tree boosting system. In *Proc. of KDD*, 2016.

C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20, 1995.

Julian Dierkes, Emma Cramer, Holger Hoos, and Sebastian Trimpe. Combining automated optimisation of hyperparameters and reward shape. *Reinforcement Learning Journal*, 3, 2024.

X. Dong and Y. Yang. NAS-Bench-201: Extending the scope of reproducible neural architecture search. In *Proc. of ICLR*, 2020.

K. Eggensperger, F. Hutter, H. Hoos, and K. Leyton-Brown. Efficient benchmarking of hyperparameter optimizers via surrogates. In *Proc. of AAAI*, 2015.

K. Eggensperger, M. Lindauer, H. Hoos, F. Hutter, and K. Leyton-Brown. Efficient benchmarking of algorithm configurators via model-based surrogates. *Machine Learning*, 107, 2018a.

K. Eggensperger, M. Lindauer, and F. Hutter. Neural networks for predicting algorithm runtime distributions. In *Proc. of IJCAI*, 2018b.

K. Eggensperger, P. Müller, N. Mallik, M. Feurer, R. Sass, A. Klein, N. Awad, M. Lindauer, and F. Hutter. HPOBench: A collection of reproducible multi-fidelity benchmark problems for HPO. In *Proc. of NeurIPS Datasets and Benchmarks Track*, 2021.

T. Eimer, M. Lindauer, and R. Raileanu. Hyperparameters in reinforcement learning and how to tune them. In *Proc. of ICML*, 2023.

J. Farebrother and P. Castro. CALE: continuous arcade learning environment. In *Proc. of NeurIPS Datasets and Benchmarks Track*, 2024.

C. Fawcett and H. Hoos. Analysing differences between algorithm configurations through ablation. *Journal of Heuristics*, 22, 2016.

E. Goan and C. Fookes. Bayesian neural networks: An introduction and survey. *Case Studies in Applied Bayesian Data Science*, 2020.

T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proc. of ICML*, 2018.

M. Henheik, T. Eimer, and M. Lindauer. Revisiting learning rate control. In *Proc. of AutoML Conf.* PMLR, 2025.

F. Hutter, H. Hoos, and K. Leyton-Brown. An efficient approach for assessing hyperparameter importance. In *Proc. of ICML*, 2014a.

F. Hutter, L. Xu, H. Hoos, and K. Leyton-Brown. Algorithm runtime prediction: Methods and evaluation. *Journal of Artificial Intelligence*, 206, 2014b.

M. Jiang, E. Grefenstette, and T. Rocktäschel. Prioritized level replay. In *Proc. of ICML*, 2021.

A. Klein, S. Falkner, J. Springenberg, and F. Hutter. Learning curve prediction with Bayesian neural networks. In *Proc. of ICLR*, 2017.

A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Y. Li, T. Rudner, and A. Wilson. A study of Bayesian neural network surrogates for Bayesian optimization. In *Proc. of ICLR*, 2024.

M. Machado, M. Bellemare, E. Talvitie, J. Veness, M. Hausknecht, and M. Bowling. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *Journal of Artificial Intelligence Research*, 61, 2018.

Y. Miao, X. Song, J. Co-Reyes, D. Peng, S. Yue, E. Brevdo, and A. Faust. Differentiable architecture search for reinforcement learning. In *Proc. of AutoML Conf.* PMLR, 2022.

V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. *arXiv:1312.5602v1 [cs.LG]*, 2013.

V. Mnih, K. Kavukcuoglu, D. Silver, A. Rusu, J. Veness, M. Bellemare, A. Graves, M. Riedmiller, A. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518, 2015.

J. Moosbauer, J. Herbinger, G. Casalicchio, M. Lindauer, and B. Bischl. Explaining hyperparameter optimization via partial dependence plots. In *Proc. of NeurIPS*, 2021.

J. Parker-Holder, V. Nguyen, and S. J. Roberts. Provably efficient online hyperparameter optimization with population-based bandits. In *Proc. of NeurIPS*, 2020.

J. Parker-Holder, R. Rajan, X. Song, A. Biedenkapp, Y. Miao, T. Eimer, B. Zhang, V. Nguyen, R. Calandra, A. Faust, F. Hutter, and M. Lindauer. Automated reinforcement learning (AutoRL): A survey and open problems. *Journal of Artificial Intelligence Research*, 74, 2022.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2011.

F. Pfisterer, L. Schneider, J. Moosbauer, M. Binder, and B. Bischl. YAHPO Gym – an efficient multi-objective multi-fidelity benchmark for hyperparameter optimization. In *Proc. of AutoML Conf.* PMLR, 2022.

S. Pineda, H. Jomaa, M. Wistuba, and J. Grabocka. HPO-B: A large-scale reproducible benchmark for black-box HPO based on OpenML. In *Proc. of NeurIPS Datasets and Benchmarks Track*, 2021.

R. Portelas, C. Colas, L. Weng, K. Hofmann, and P. Oudeyer. Automatic curriculum learning for deep RL: A short survey. In *Proc. of IJCAI*, 2020.

Y. Pushak and H. Hoos. Algorithm configuration landscapes: - more benign than expected? In *Proc. of PPSN*, 2018.

Y. Pushak and H. Hoos. AutoML loss landscapes. *ACM Transactions on Evolutionary Learning and Optimization*, 2, 2022.

C. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

R. Sass, E. Bergman, A. Biedenkapp, F. Hutter, and M. Lindauer. Deepcave: An interactive analysis tool for automated machine learning. In *ICML ReALML Workshop*, 2022.

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

G. Shala, S. P. Arango, A. Biedenkapp, F. Hutter, and J. Grabocka. HPO-RL-Bench: A zero-cost benchmark for HPO in reinforcement learning. In *Proc. of AutoML Conf.* PMLR, 2024.

J. Vanschoren, J. van Rijn, B. Bischl, and L. Torgo. OpenML: Networked science in machine learning. *SIGKDD Explor. Newsl.*, 15, 2014.

Z. Wang, G. Dahl, K. Swersky, C. Lee, Z. Nado, J. Gilmer, J. Snoek, and Z. Ghahramani. Pre-trained Gaussian processes for Bayesian optimization. *Journal of Machine Learning Research*, 25, 2024.

M. Wever, M. Muschalik, F. Fumagalli, and M. Lindauer. HyperSHAP: Shapley values and interactions for hyperparameter importance. *CoRR*, abs/2502.01276, 2025.

C. Ying, A. Klein, E. Christiansen, E. Real, K. Murphy, and F. Hutter. NAS-Bench-101: Towards reproducible Neural Architecture Search. In *Proc. of ICML*, 2019.

A. Zela, J. Siems, L. Zimmer, J. Lukasik, M. Keuper, and F. Hutter. Surrogate NAS benchmarks: Going beyond the limited search spaces of tabular NAS benchmarks. In *Proc. of ICLR*, 2022.

L. Zimmer, M. Lindauer, and F. Hutter. Auto-Pytorch: Multi-fidelity metalearning for efficient and robust AutoDL. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43, 2021.

Hyperparameter	LCBench
batch size	$\log([16, 512])$
learning rate	$\log([10^{-4}, 0.1])$
momentum	$[0.1, 0.99]$
weight decay	$[10^{-5}, 0.1]$
num layers	$\{1, 2, 3, 4\}$
max units	$\log([64, 1024])$
max dropout	$[0.0, 1.0]$

Table 4: The hyperparameter search space of the landscapes in the *LCBench*

A Datasets

All the datasets used for training the surrogate models have been obtained by a random search in the corresponding configuration spaces of each dataset. The configuration spaces of the RL algorithms PPO [Schulman et al., 2017], DQN [Mnih et al., 2013] and SAC [Haarnoja et al., 2018] used in the *ARLBench* [Becktepe et al., 2024] are given in Table 1, 2 and 3, respectively.

Table 1: The hyperparameter search space of *ARLBench* for PPO.The first column specifies the different classes of tasks in the benchmark that use different search spaces.

Table 2: The hyperparameter search space of *ARLBench* for DQN. The target update interval is a conditional hyperparameter that is only optimised when a target network is used. Similarly, α, β and ϵ of the buffer are only optimised when priority sampling is used. The first column specifies the different classes of tasks in the benchmark that use different search spaces.

Table 3: The hyperparameter search space of *ARLBench* for SAC. The hyperparameter search space for SAC. The hyperparameter tau is a conditional parameter that is only optimised when a target network is used. Similarly, α, β and ϵ of the buffer are only optimised when priority sampling is used. The first column specifies the different classes of tasks in the benchmark that use different search spaces.

The search spaces for *LCBench* [Zimmer et al., 2021] are given in Table 4 and the search space for *PD1* [Wang et al., 2024] in Table 5.

For *LCBench* and *PD1*, performance was measured in classification accuracy and normalised as such. For *ARLBench* and *Lunar Lander 2048*, we used theoretical minima and maxima if available and otherwise used the performance of a random policy as minima and strong baselines from the literature as maximal values. The specific values for normalisation are given in Table 6.

Table 6: Minimal and maximal performance values used for normalising the RL performance landscapes. If the performance value in a landscape was greater or smaller than the normalisation performance, the minimal or maximal performance value of the landscape was used for normalisation.

Hyperparameter	Transformer	Wide-ResNet	XFormer
batch size	2048	$\{256, 2048\}$	64
learning rate decay	$[0.010543, 0.9885653]$		
learning rate initial	$\log([10^{-5}, 9.986256])$		
learning rate power	$[0.100811, 1.999659]$		
momentum	$[5.9 \cdot 10^{-5}, 0.9989986]$		

Table 5: The hyperparameter search space of the landscapes in *PD1*. The *Transformer* and *ResNet* architectures have been sampled with two categorical choices for the batch size, whereas the batch size for the *XFormer* was fixed.

Whenever possible, theoretical minimal and maximal values have been used. For environments from the ALE, we used the random and human scores presented by Mnih et al. [2015]. For the maximum scores in Box2D LunarLander and Brax Ant and Humanoid, we used the best results from Dierkes et al. [2024], while for Brax Hopper and HalfCheetah, we relied on the top performances reported by **?**. The lower values of -200 and -2000 for the Box2D and Brax environments have been chosen similarly to *ARLBench* due to reported numerical instabilities. If the performance value in a landscape was greater or smaller than the normalisation performance, the respective landscape performance was used for normalisation.

B Surrogate Tuning

In Table 7, we depict the search spaces for the RF, XGB, GP and SVM surrogate models.

Hyperparameter	Gaussian Process
kernel	{RBF, Matern-0.5, Matern-1.5, Matern-2.5}
alpha	$[10^{-12}, 10^{-5}]$

Hyperparameter	Random Forest
number estimators	$\log([16, 128])$
minimal samples split	$[2, 20]$
minimal samples leaf	$[1, 20]$
maximal features	$[0.1, 1.0]$
bootstrap	True, False

Hyperparameter	SVM
C	$\log([1, 20])$
coef0	$[-0.5, 0.5]$
degree	$\log([1, 128])$
epsilon	$\log([0.1, 0.99])$
gamma	{scale, auto}
kernel	{linear, ref, poly, sigmoid}
shrinking	{True, False}
tol	$\log([10^{-4}, 0.01])$

Hyperparameter	XGB
max depth	$[1, 20]$
minimal child weight	$\log([1, 100])$
colsample byte	$[0.0, 1.0]$
colsample bilevel	$[0.0, 1.0]$
lambda	$\log([0.001, 1000])$
alpha	$\log([0.001, 1000])$
learning rate	$\log([0.001, 0.1])$

Table 7: The search spaces for our different surrogate models.

The most common performance objective of RL is the final performance of a training run. However, alternative metrics like the maximal performance over the full training run or the area under the

curve of the learning curve can be used as well to assess the performance of a configuration. We trained surrogate models for predicting all three performance metrics to better understand how the performance metric impacts surrogate model quality. The comparison for *ARLBench* and *LunarLander 2048* is depicted in Figure 7.

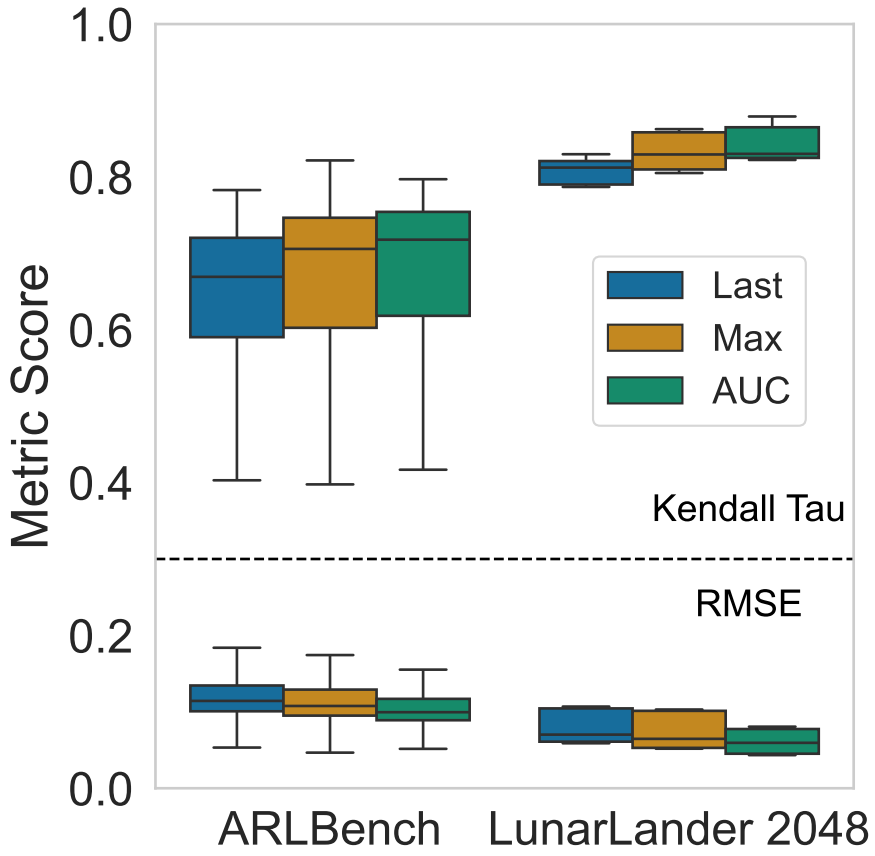

Figure 7: The Boxplots compare the quality of a surrogate model measured in the Kendall Tau and RMSE of its predictions. We consistently observe that the area under the curve results in the best surrogate models, followed by the maximal performance and the final performance.

We consistently observe that the area under the curve results in the best surrogate models, followed by the maximal performance and the final performance. However, the results do not appear to be substantially different. In particular, on the *ARLBench*, we still observe a large performance spread over the different environments.

C Local Parameter Importance

To estimate the influence of individual hyperparameters on performance, we compute the *Local Performance Importance (LPI)* by measuring the performance variance across configurations that differ only in a single hyperparameter $h \in \mathcal{H}$. Let σ_h^2 denote the variance in performance when varying h while keeping all other hyperparameters fixed. These variances are then normalised by the total variance across all hyperparameters to yield relative importance scores:

$$\text{LPI}_h = \begin{cases} \frac{\sigma_h^2}{\sum_{h' \in \mathcal{H}} \sigma_{h'}^2} & \text{if } \sum_{h'} \sigma_{h'}^2 > 0 \\ 0 & \text{otherwise} \end{cases}$$

The resulting LPI values indicate the proportion of total performance variation that can be attributed to each hyperparameter. Higher values suggest greater local influence on performance.

In Figure 8, we show the raw results of the LPI for PPO. For the 25th percentile, we can observe that the surrogate models do seem to have problems predicting LPIs with long tails correctly. Furthermore, the surrogate models do not correctly assess the equal importance of the hyperparameters and significantly overestimate some hyperparameters. Similarly, in the 50th percentile, the learning rate is significantly overestimated while the importance of the normalisation advantage is not captured accurately.

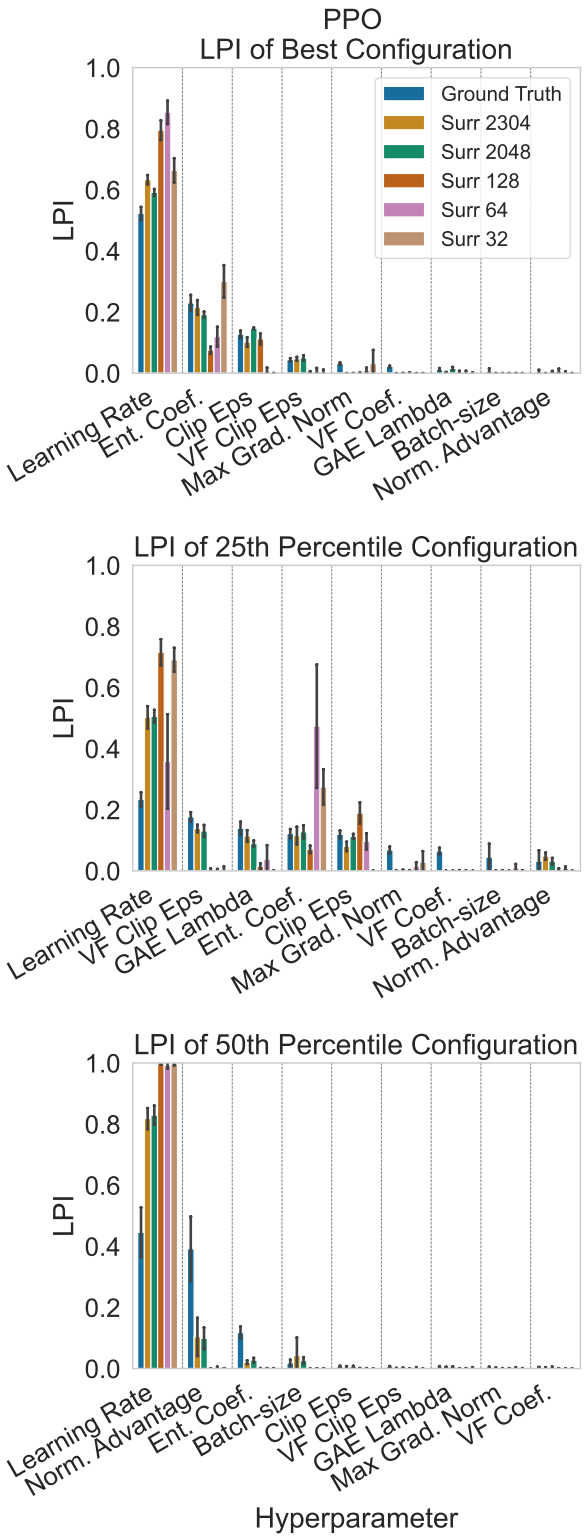

Figure 8: PPO LPIs of different quantile configurations

D Ablation Paths

An *ablation path* is a sequence of configurations in which hyperparameters are incrementally set to values from a reference configuration (e.g., the best-performing one). Starting from a baseline (e.g., a default or random configuration), one hyperparameter is changed at a time, and its impact on performance is recorded. At each step, the hyperparameter whose change yields the highest performance improvement is selected. This greedy process continues until the full reference configuration is reconstructed. Ablation paths help to understand the contribution of individual hyperparameters and their interactions along a trajectory toward optimal performance.

To evaluate the performance of our surrogate models on ablation paths, we measured their ability to identify the most influential hyperparameter whenever the ground-truth ablation path exhibits a performance change greater than 10Figure 9 presents the top-1, top-3, and top-5 accuracy of the surrogate models.

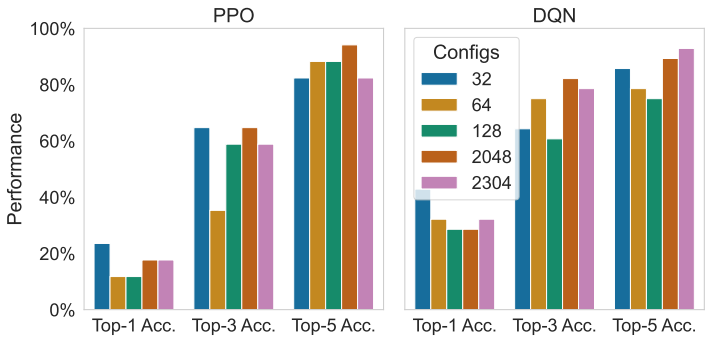

Figure 9: The accuracy of the surrogate model of predicting the most significant hyperparameter whenever a performance shift or more than 10% occurs in the ground-truth ablation paths.

A top-1 accuracy of 20% means that in only 20% of the cases where the ground-truth shows a significant performance shift, the surrogate correctly identifies the responsible hyperparameter. This is a notably low accuracy, especially considering that altering a single hyperparameter in an ablation path can drastically change subsequent steps. Such results suggest that the surrogate models are not reliably capturing the underlying dependencies. Furthermore, we can not observe an increased performance when using more configurations to train the surrogate models. Ablation paths commonly consist of many high-performing configurations and we previously observed that our surrogate models perform especially badly in high-performing regions of the landscape, which are sampled scarly. Therefore, we hypothesise that, in particular, more high-performing configurations are necessary to train well-performing surrogate models.

To verify that the low top-1 accuracy is not merely due to noise in the configurations, we report the average rank distance between the surrogate prediction and the true most important hyperparameter in Figure 10. Additionally, Figure 10 presents the average absolute performance difference between the surrogate prediction and the best-performing configuration.

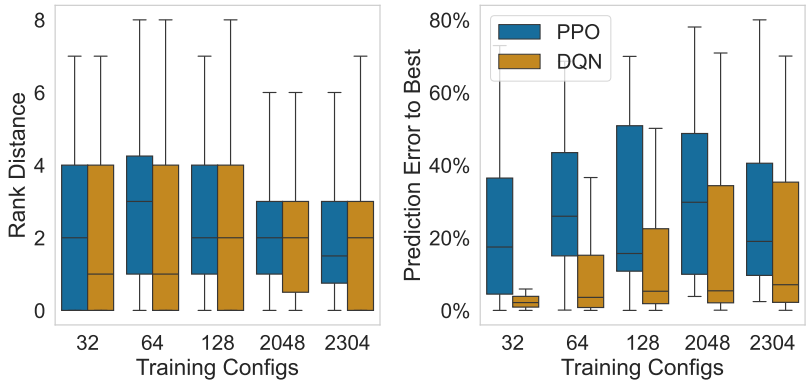

Figure 10: The left figure depicts the rank distance between the ground-truth's most important hyperparameter and the surrogate model's top-ranked prediction. The right figure depicts the percentage of the absolute difference between the surrogate's ground-truth prediction and the top-ranked prediction.

For PPO, we observe a median rank distance of approximately 2 and an average absolute performance difference of around 20%. In particular, the latter value indicates that the worse performance of the surrogate models can not be explained merely by noise. In the case of DQN, these differences appear much smaller; however, further analysis revealed that the surrogate models assign uniformly low performance scores to most configurations. This behaviour, while yielding low numerical differences, is also undesirable as it indicates a lack of meaningful discrimination across configurations.

In Figure 11, we show the raw results of the ablation paths for PPO on different percentiles as source configs in the *LunarLander 2048* landscapes. The target configuration was always the best-performing configuration in the landscape.

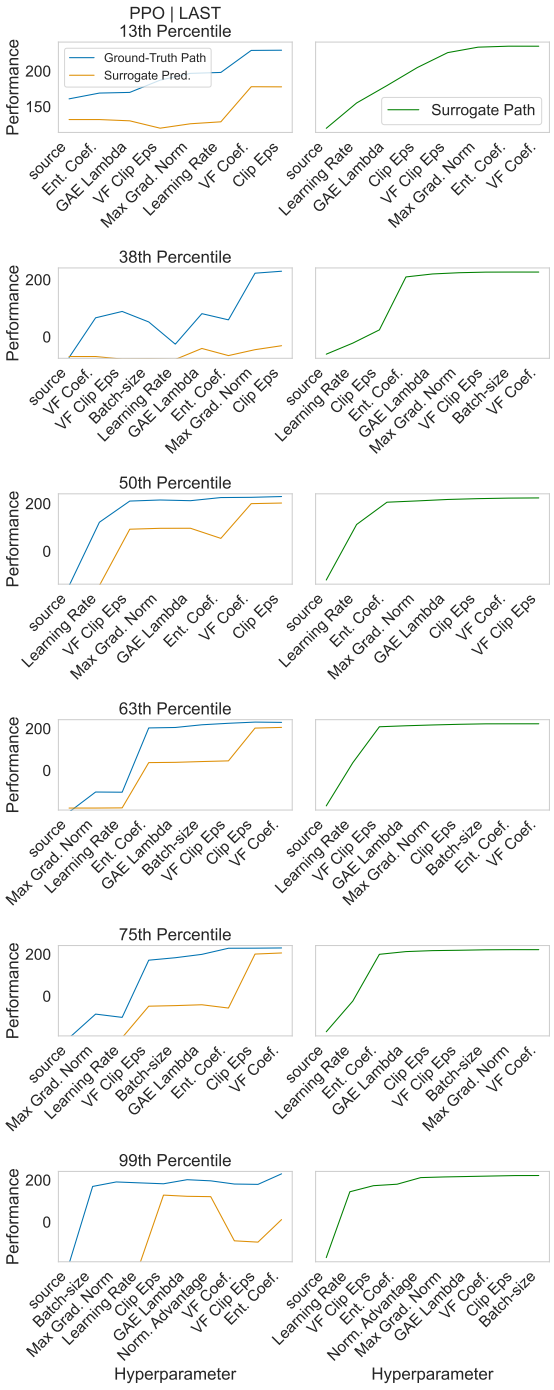

Figure 11: Ablation Path results for different percentile configurations on the PPO LunarLander landscape. The left column shows the ground-truth ablation paths in blue with the respective performance prediction of the random forest surrogate model in orange. The right column shows the resulting ablation path when using the surrogate model predictions.

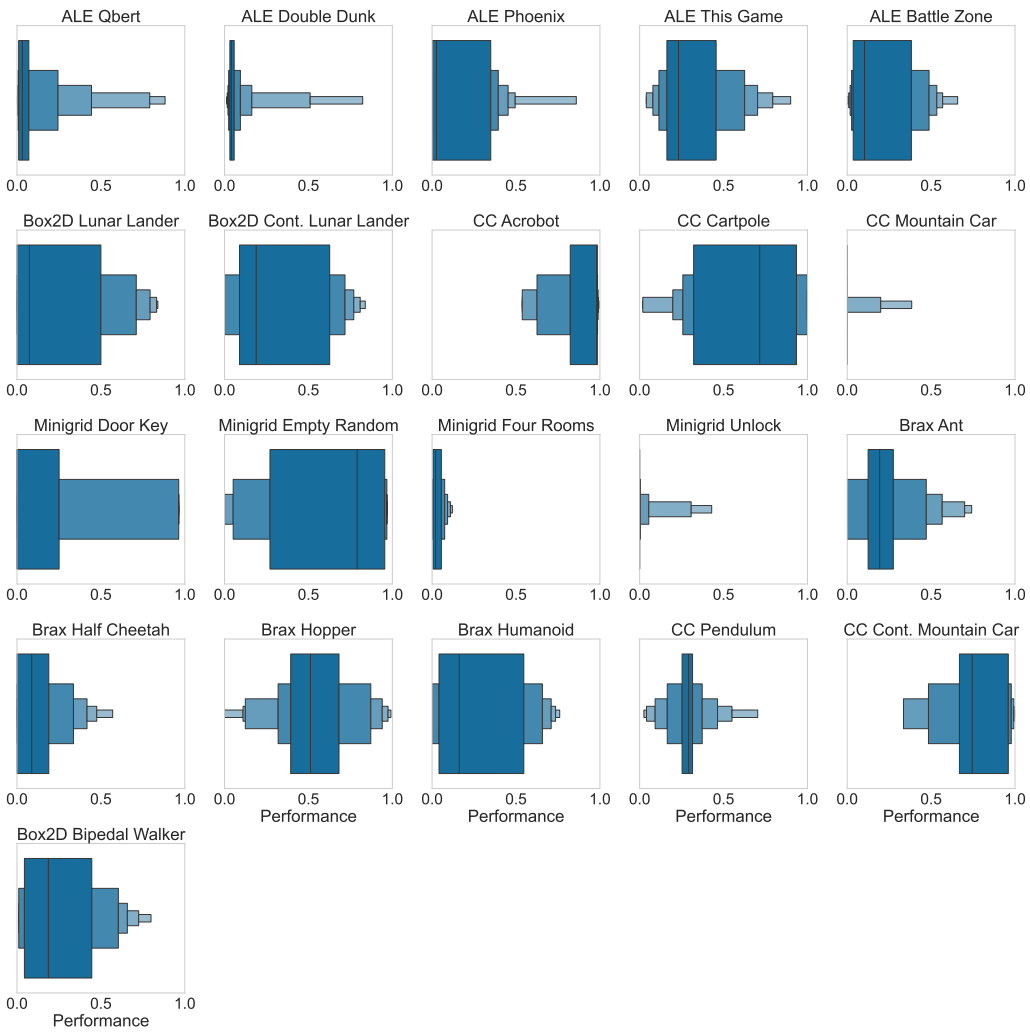

Figure 12: ARLBench PPO environments with clearly defined minimum and maximum have a much greater spread over all environments.

E Performance Spreads

In Figures 12, 13, 14 and 15, we show the full performance spread over all the different tasks in the *LCBench* and *ARLBench*.

F CI Scatter Plots

In Figure 18 and 19, we show the scatter plots between performance and confidence intervals over all the different tasks in the *ARLBench* and *LCBench*.

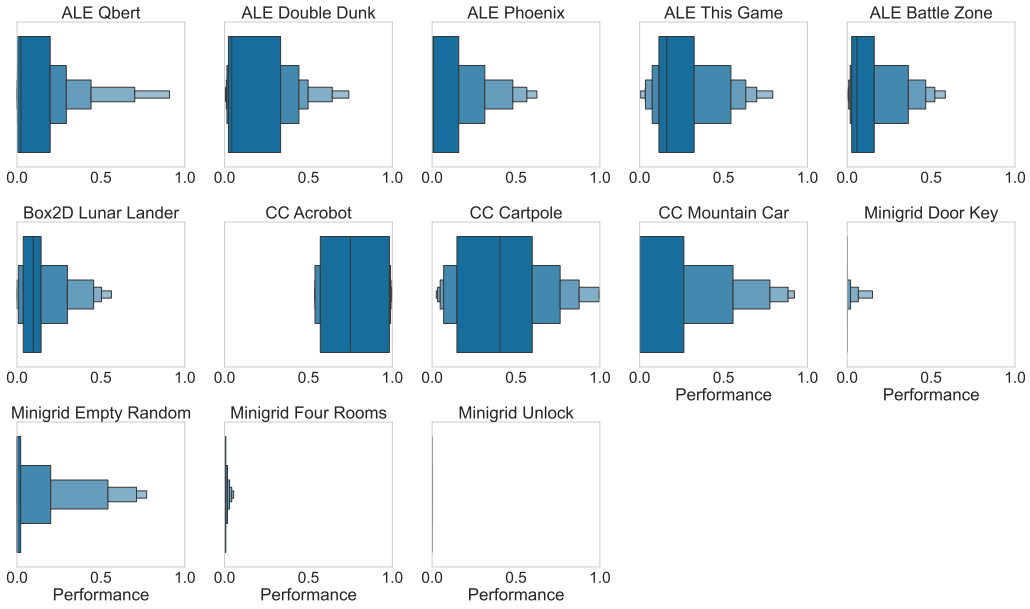

Figure 13: ARLBench DQN environments with clearly defined minimum and maximum have a much greater spread over all environments.

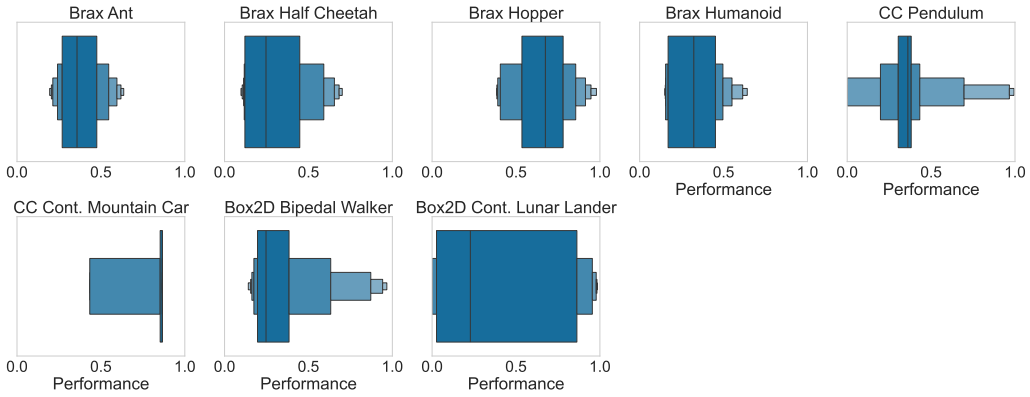

Figure 14: ARLBench SAC environments with clearly defined minimum and maximum have a much greater spread over all environments.

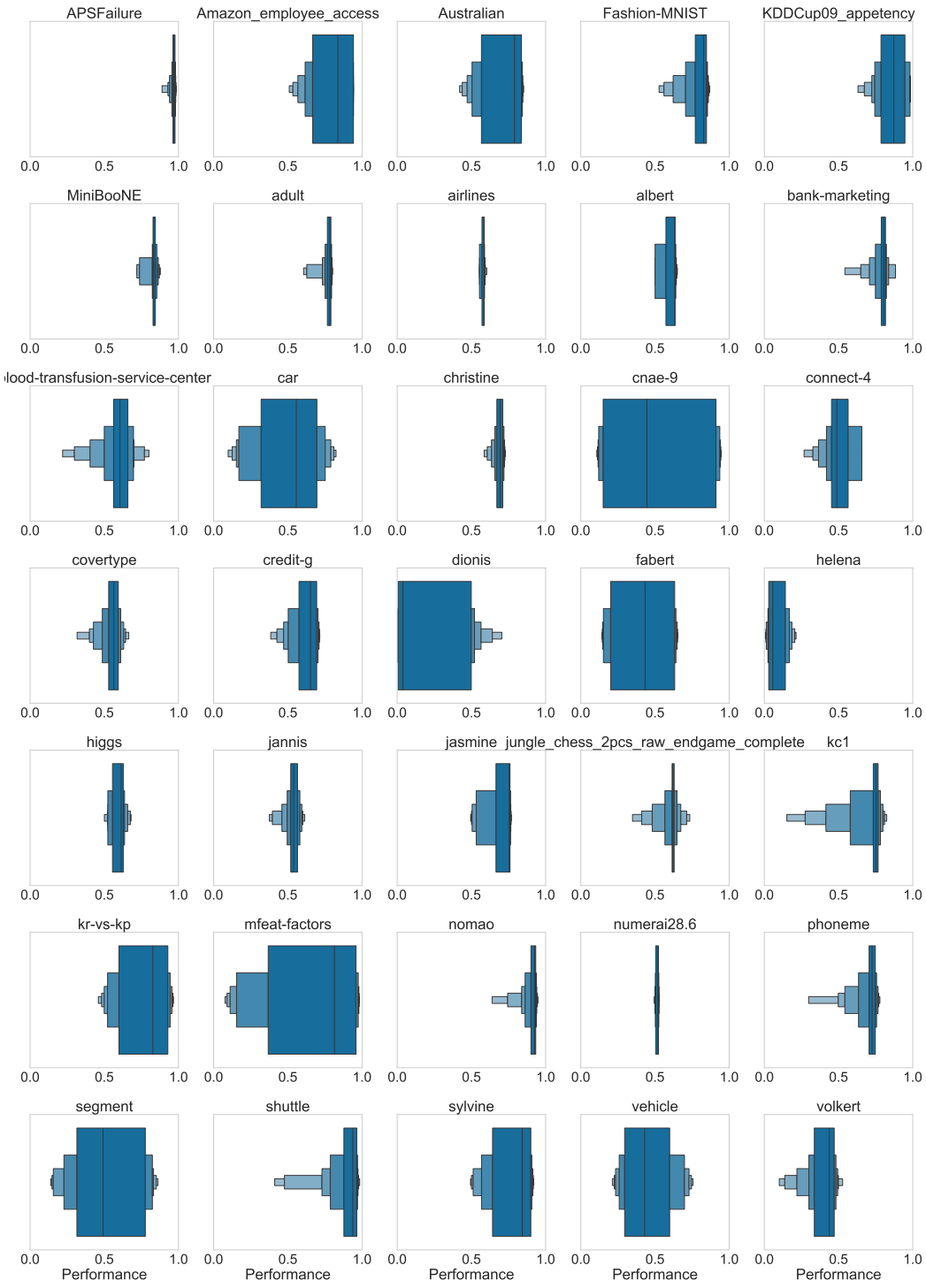

Figure 15: LCBench performance spread is quite tight in many environments. Therefore, a rather simple benchmark, as all hyperparameters perform similarly.

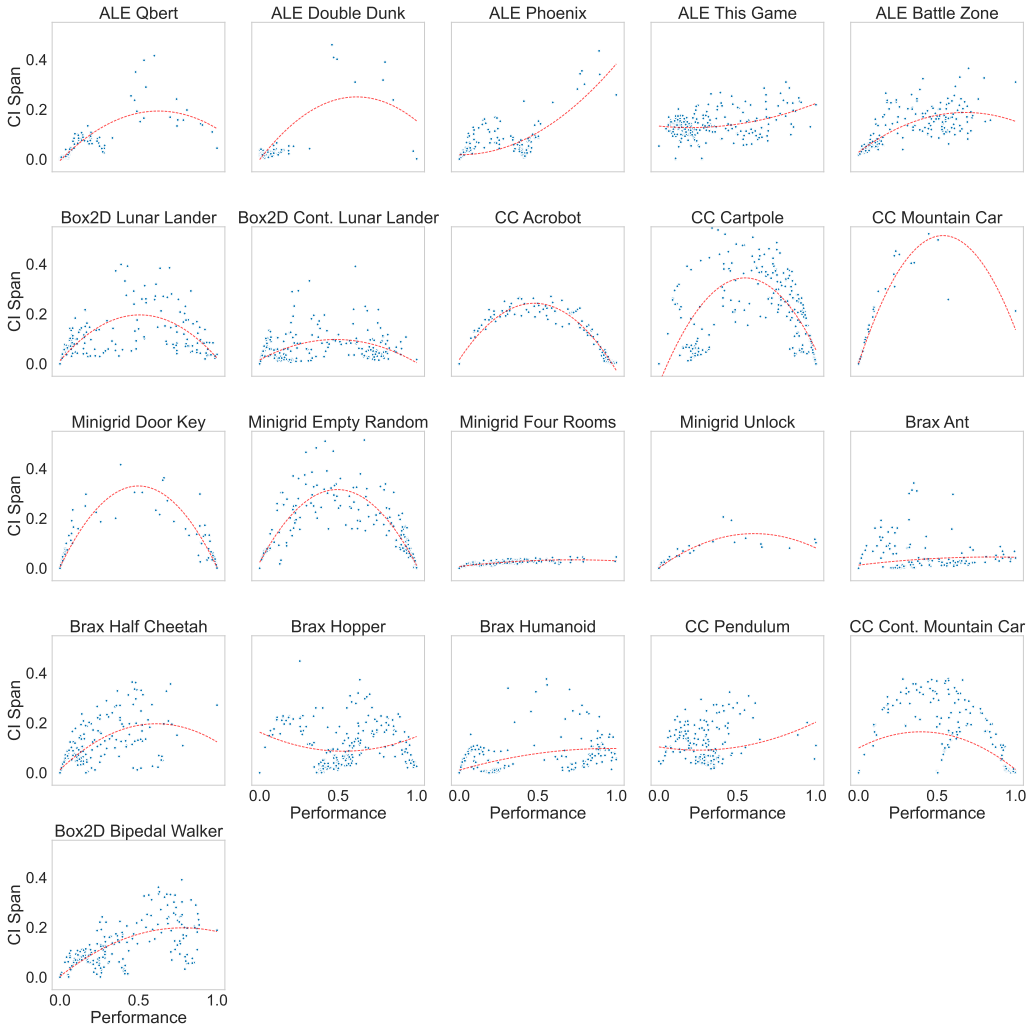

Figure 16: ARLBench PPO ci/performance scatter plots. The red line shows the best fit of a polynomial of degree two.

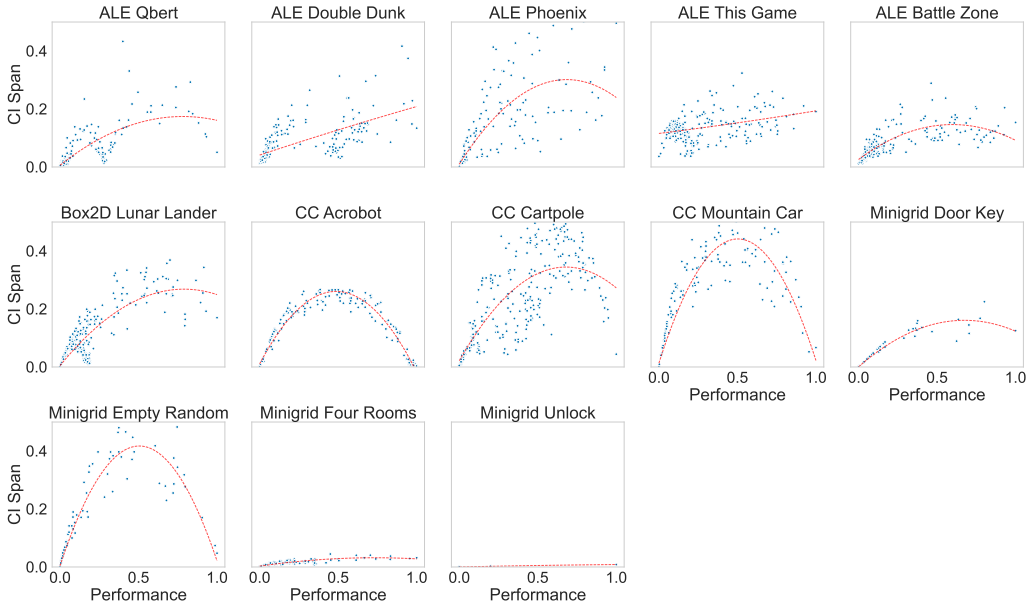

Figure 17: ARLBench DQN ci/performance scatter plots. The red line shows the best fit of a polynomial of degree two.

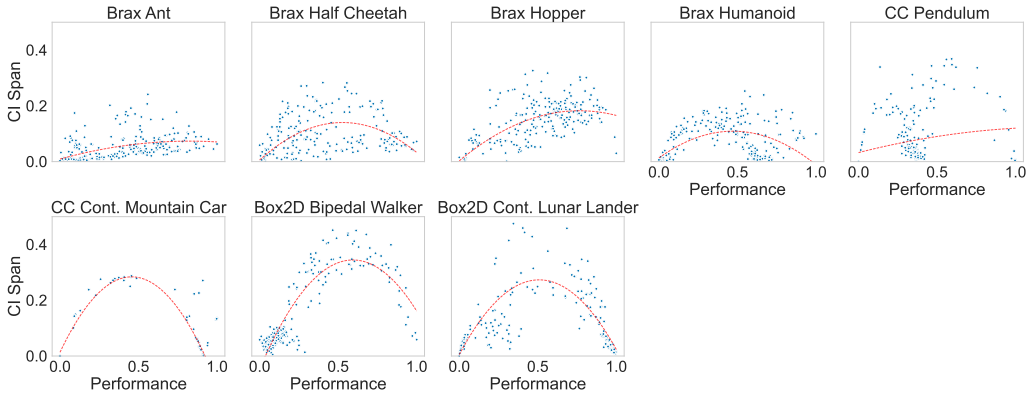

Figure 18: ARLBench SAC ci/performance scatter plots. The red line shows the best fit of a polynomial of degree two.

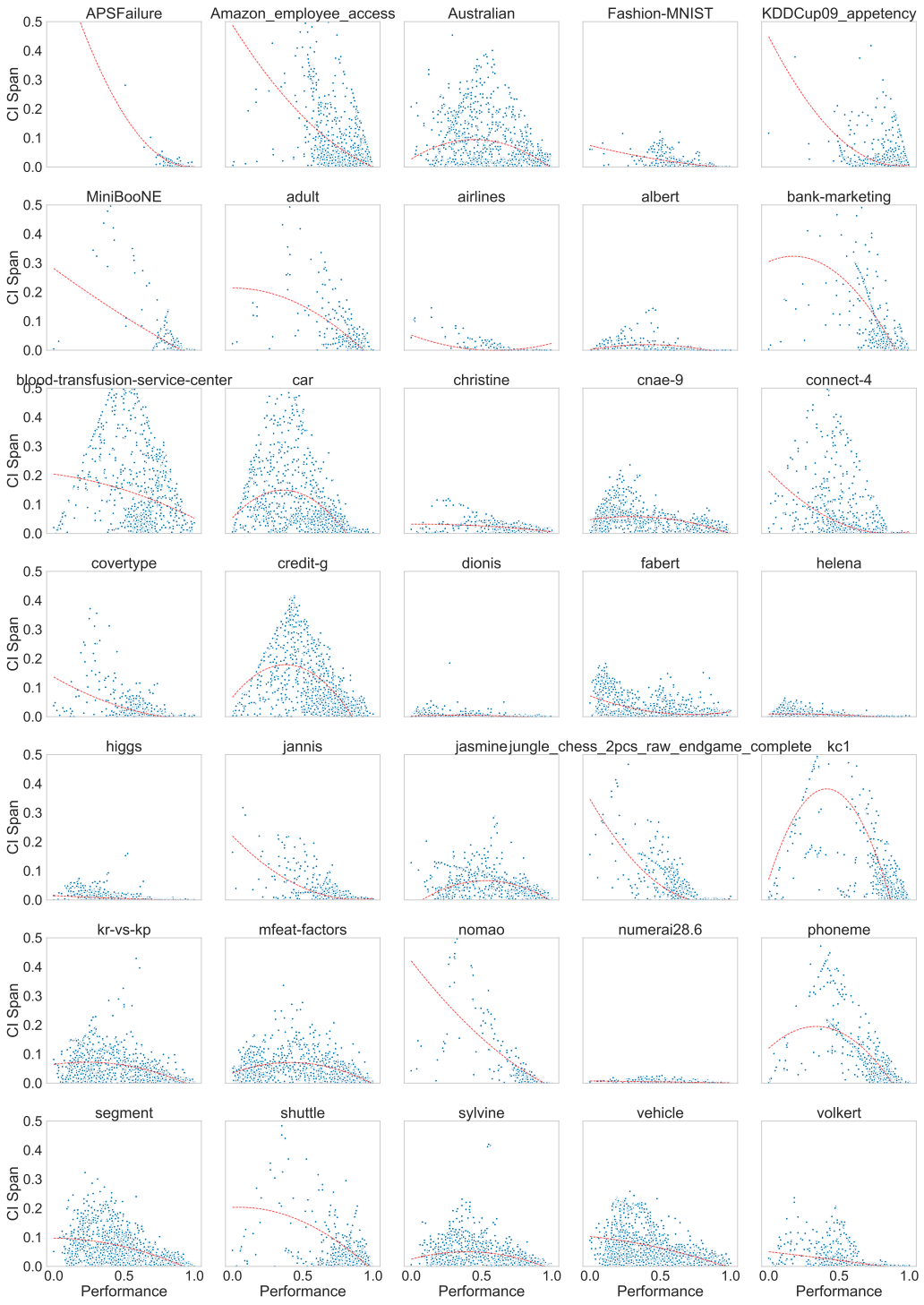

Figure 19: LCBench ci/performance scatter plots. The red line shows the best fit of a polynomial of degree two.