AN EMPIRICAL STUDY OF REINFORCEMENT LEARNING SCALING LAWS

Anonymous authors
Paper under double-blind review

ABSTRACT

The discovery of scaling laws enables the prediction of *the performance*, *the optimal resource allocation strategy*, and *the best hyperparameter configuration* of a large-scale training job based on inexpensive experiments at smaller scales. Guided by these empirical scaling laws, the deep learning community has achieved great success in training large (self-)supervised models on Internet-scale datasets. In contrast, despite its potential and wide application, reinforcement learning (RL) has yet to find a guiding principle for predictable scaling. In this work, we conduct an empirical study of RL scaling laws of an actor-critic algorithm in two procedurally generated environments, Starpilot and Craftax. On the one hand, we discovered empirical scaling laws with strong predictive power in both environments. The discovered scaling laws accurately predict the aforementioned three quantities for larger-scale training jobs up to two orders of magnitude in total training compute. In particular, we employ a variant of Elo scores for the single-agent RL paradigm as an alternative performance metric than the commonly used mean episode return. Our results show that the Elo score is a much more predictable metric than the mean episode return, making it a better choice for fitting scaling laws. By applying our discovered scaling laws to increase training compute, we established a new state of the art in the challenging Craftax environment. On the other hand, beneath the surface of the global scaling trend, we observed a non-monotonic scaling pattern of the compute-optimal model size in Craftax. We connect this observation to a pathology known as Ray interference.

1 INTRODUCTION

The discovery of empirical scaling laws has revolutionized deep learning, in particular for generative modeling, by transforming model development from a process of heuristic trial-and-error into a *predictable* engineering discipline (Kaplan et al., 2020; Hoffmann et al., 2022; Bi et al., 2024). Before launching a large-scale training job with a compute budget of C FLOPs, three critical questions need to be answered (Muennighoff et al., 2023): (i) **(Performance)** What is the best achievable performance? (ii) **(Allocation)** What are the optimal model and dataset sizes for achieving peak performance? (iii) **(Hyperparameter)** What are the optimal hyperparameters for the training job? Scaling laws provide a tool for researchers to answer these questions by conducting inexpensive experiments at smaller scales and extrapolating to larger scales.

In contrast to the success of scaling in (self-)supervised learning, reinforcement learning (RL) has yet to find its own guiding principles for predictive scaling. While scaling has powered landmark achievements from game-playing agents (Silver et al., 2017; Vinyals et al., 2019; Berner et al., 2019), to fine-tuning large language models (Guo et al., 2025; Khatri et al., 2026), the foundational scaling dynamics of RL remain poorly understood. Unlike (self-)supervised learning, where the data distribution is stationary, RL agents learn from experiences that continuously change as their own policy evolves. Learning in RL is thus inherently non-stationary. This active coupling between policy improvement and data generation implies that RL may be governed by scaling laws distinct from those of passive, supervised regimes. Moreover, unlike the cross entropy loss which has a clear information-theoretical grounding, rewards in RL are often designed in an ad-hoc manner based on heuristics or domain knowledge. Consequently, the common RL performance metric, mean episode return, may not vary smoothly (Hilton et al., 2023). Due to these differences, it is unclear if the same conclusion, and predictive power, from the (self-)supervised learning scaling laws apply to RL.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Existing work on RL scaling laws primarily answer only a subset of the aforementioned three questions while also lacking validation of the prediction fidelity of the fitted laws (Jones, 2021; Hilton et al., 2023; Neumann & Gros, 2023). We argue that the most critical value of scaling laws is in their predictive power and thus validating the predictions is a centerpiece of scaling law studies. In this paper, we present an empirical study of scaling laws in RL. We study all three questions with validation of the prediction fidelity from small-scale experiments to large-scale experiments in each case. To achieve this, we conducted experiments in two procedurally generated environments, Starpilot from the ProcGen suite (Cobbe et al., 2020) and Craftax (Matthews et al., 2024). We fit scaling laws for the optimal hyperparameters, model and data sizes, and performance as a function of total training FLOPs C. We validated the predictive power of the fitted laws by testing their prediction accuracy at one or two orders of magnitude larger training FLOPs. To overcome the non-smoothness of the mean episode return, we employ a variant of the Elo score (Elo, 1978) for the single-agent RL paradigm and showed that it extrapolates more effectively whereas the mean episode return does not, making it a critical choice for scaling studies. Our analysis in Craftax reveals that, despite the empirical accuracy of the fitted laws, the compute-optimal frontier does not follow a single power law across the entire training process. We connect this empirical observation to a phenomenon known as the Ray interference (Schaul et al., 2019). Based on this observation, we argue against the existence of power laws for arbitrary RL environments.

Our main contributions are:

- We establish a two-stage empirical framework for studying RL scaling laws. Our experimental results show that optimal model size and key hyperparameters (e.g., batch size and learning rate) follow predictable power-law relationships with respect to training compute.
- We address the non-smoothness of mean episode return and show that a single-agent Elo metric scales log-linearly as a function of training compute, enabling reliable performance extrapolation up to two orders of magnitude.
- Guided by our scaling laws, we train an actor-critic agent to $1\text{e}19$ FLOPs and set a new state-of-the-art performance of 52.1% on the challenging Craftax benchmark.
- We identify a local inverse scaling pattern in Craftax and connect it to "Ray interference," showing how global trends can mask non-monotonic local dynamics.

2 Related Work

In the past decade, empirical power laws were discovered in multiple deep learning application domains including image classification (Hestness et al., 2017; Rosenfeld et al., 2020), image generation (Henighan et al., 2020), and language modeling (Kaplan et al., 2020; Hoffmann et al., 2022). Kaplan et al. (2020) observed that the cross entropy loss of an autoregressive language model scales as a power law with the model size, the dataset size, and the amount of compute used for training across several orders of magnitude. Their work provides a method to answer the *performance* question. Based on their fitted scaling laws, Kaplan et al. (2020) recommended scaling model size more aggressively than dataset size as training compute increases. Following Kaplan et al. (2020), Hoffmann et al. (2022) conducted a study on the optimal model size for a certain training compute budget, directly addressing the *allocation* question. Contradictory to Kaplan et al. (2020), their work showed that the model size and dataset size should grow at an equal pace while scaling up training compute. In both works, the power laws were fitted to explain the observed data. Their predictions were not validated at larger scales. Moreover, many experimental details such as hyperparameter choices are missing or poorly studied, leaving the *hyperparameter* question unanswered. To address these limitations, DeepSeek (Bi et al., 2024) studied all three questions of performance, allocation, and hyperparameter and tested the predicive accuracy of the fitted scaling laws at larger scales. Our work builds upon these works. We employed their general methodology to study all three questions and tested the predictive accuracy of the corresponding scaling laws. Meanwhile, our work is different from these works as we focus on RL instead of language modeling.

There are few existing works on RL scaling laws. The most closely related work is by Hilton et al. (2023), who studied scaling laws of single-agent RL. They identified the difficulty of predicting the mean episode return at larger training compute due to its non-smoothness. To address this challenge, they proposed a new metric called *intrinsic performance* which measures the performance of an RL

agent by the minimum amount of training compute needed to achieve parity with it. Although by definition intrinsic performance is not suited for answering the performance question, it served as a tool to answer the allocation question. Fu et al. (2026) studied scaling laws for value-based RL algorithms with a focus on the allocation question. Jones (2021) and Neumann & Gros (2023) studied scaling laws of the AlphaZero agent (Silver et al., 2017) in the context of two-player zero-sum games. Both works found that the Elo score (Elo, 1978) of an agent scales predictably with the amount of training compute. Jones (2021) further showed that the Elo scaling law itself scales with the size of the game. Like Hilton et al. (2023), we focus on the single-agent RL setup. But instead of using intrinsic performance, we employ a novel Elo variant as the performance metric which allows us to answer the performance question. The idea of using Elo to measure agent performance is inspired by Jones (2021); Neumann & Gros (2023). Moreover, these works only studied a subset of the three scaling law questions. To the best of our knowledge, our work is the first to study all three questions in the context of RL, rendering a more complete picture of RL scaling laws.

Recently, the success of reasoning models (Matthews et al., 2024; Comanici et al., 2025; Guo et al., 2025) reinvigorated interest in RL scaling laws in the context of large language model (LLM) fine-tuning. Examples include Khatri et al. (2026), Cheng et al. (2026), and Tan et al. (2026). These works focus on a specific application domain of RL and concern domain-specific quantities such as the number of parallel rollouts per prompt and the pass rate on a specific dataset. In contrast, our work focuses on the more general sequential-decision making setup with minimum assumptions about the environment and the reward structures.

3 METHOD

Our methodology implements a two-stage framework adapted from the paradigm proposed by Bi et al. (2024). First, we characterize hyperparameter scaling dynamics by conducting a systematic grid search across small-to-medium compute budgets to establish empirical power laws that parameterize optimal configurations as functions of training compute. Second, we determine the joint scaling behavior of model size and performance using the isoFLOPs profile approach (Hoffmann et al., 2022), dynamically configuring the hyperparameters in this stage using the optimal values prescribed by our first-stage scaling laws. To validate this framework, we first conduct a rigorous scientific evaluation of prediction fidelity at a $10\times$ compute frontier by independently comparing each scaling law's prediction against the empirical ground truth. We then showcase the practical utility of these scaling laws at a $100\times$ compute scale, mirroring real-world constraints where validating prediction accuracy is computationally prohibitive. Rather than evaluating the laws themselves, this step demonstrates their use in practice: we execute a large-scale training run configured by our hyperparameter and model size prescriptions, comparing its realized performance to the performance scaling law's prediction to verify that the high-budget training job meets expectation.

3.1 HYPERPARAMETER SCALING LAWS

We use the A2C algorithm (Mnih et al., 2016) across all experiments in this work. In addition to batch size and learning rate, the entropy regularization strength is also an importance hyperparameter in A2C. We did a preliminary grid search over combinations of these three hyperparameters at a low compute regime. The optimal entropy regularization strength is consistent across different compute budgets. Following this observation, we decided to focus on studying the scaling dynamics of batch size and learning rate. Specifically, we fixed the rollout length of each trajectory and varied batch size by changing the number of parallel environment instances.

To characterize the scaling dynamics of hyperparamters, we first need to find their optimal values for a set of different training budgets at a small scale. For each training FLOPs C, we conduct a grid search on batch size and learning rate across a few different model sizes to identify their optimal values. Due to randomness in the training process, the empirical performance is a noisy estimate of the true performance. Thus the empirical optimal hyperparameter configuration is often not the true optimum. RL performance evaluation is known to suffer from higher variance than supervised learning (Henderson et al., 2018), which makes the identification problem even harder. To reduce the variance of the empirical performance measurement, we conduct multiple independent jobs for each configuration with different random seeds. We then aggregate the episode returns across random seeds by their interquartile mean (IQM). Moreover, to take into account the randomness of the

empirical results, we do bootstrapping by resampling the random seeds for the experiments. For each resampling, we pick the batch size and learning rate that yields the highest IQM of the episode return. We call this configuration a "competitive" configuration. Through bootstrapping, we obtain a set of competitive configurations for each training compute C. Instead of fitting the empirically optimal configurations, we fit power laws to all competitive configurations in the following forms:

$$B^*(C) = a_B C^{d_B} \tag{1}$$

$$\alpha^*(C) = a_\alpha C^{d_\alpha} \tag{2}$$

$B^*(C)$ and $\alpha^*(C)$ represent the average competitive batch size and learning rate at training FLOPs C respectively. a_B, d_B, a_α, and d_α are parameters fitted from the data.

Bi et al. (2024) employ a method that shares the same spirit as ours to combat the noise in optimal hyperparameter identification. They define "near-optimal" hyperparameters as the ones that yield a loss with in 0.25% of the empirical minimum. Similar to our method, they then fit the scaling laws to the near-optimal hyperparameters. We do not use the same method because unlike cross entropy the episode return needs not vary smoothly through training. It is hard to pick a threshold for the "near-optimal" definition. Instead, we rely on bootstrapping to address randomness in experiments.

3.2 Allocation scaling laws

To understand the optimal allocation of training compute, we first need to calculate the total FLOPs C of a training run. In Transformer-based language models, this is often approximated by $C = 6ND$ where N denotes the total number of parameters in the model and D denotes the total number of tokens consumed during training (Kaplan et al., 2020; Hoffmann et al., 2022). This approximation is based on dense layers. However, we use convolutional networks which does not follow the same calculation. Therefore, similar to the FLOPs/token in (Bi et al., 2024), we use the notion of FLOPs-per-frame (FPF), denoted by M, as a measurement of the model size. Here a frame refers to the observation of a single step in the environment. FPF is defined as the number of FLOPs spent on each observation during training. The total training compute is thus $C = MD$ where D denotes the number of frames the agent is trained for. Note that for a model-based RL agent we may need to take additional factors such as planning steps into account. We focus on model-free RL agents in this work and leave the study of model-based agents to future work.

With the clear factoring of training FLOPs, we used the isoFLOPs profile approach (Hoffmann et al., 2022) to study the optimal compute allocation between model size and data size. We need to find the optimal allocation at a set of different training FLOPs at small scale. For each training FLOPs C, we trained models in different sizes using hyperparameters predicted by their corresponding scaling laws. To combat variance, we run multiple independent jobs for each model size with different random seeds and aggregate the results by IQM. Then we fit a parabola to approximate the mapping from FPF to performance. We extract the optimum of the parabola to identify the opitmal FPF M^* for training FLOPs C. Lastly, we fit a power law in the following form:

$$M^*(C) = a_M C^{d_M} \tag{3}$$

a_M and d_M are parameters fitted from data. We do not need to fit a separate law for the optimal data size N^* as it can be derived from M^*.

3.3 Performance scaling laws in Elo

Before predicting the performance as a function of the training compute, we must first determine how performance is measured. As observed in (Hilton et al., 2023), the compute-optimal frontier need not vary smoothly when the performance is measured in mean episode return. The fundamental reason is that the magnitude of rewards (and thus returns) does not reflect the learning progress smoothly. To address this problem, we employ a variant of the Elo score (Elo, 1978) as a performance measurement of a *single-agent* RL training system. Elo is typically used to measure the relative playing strength of different players within a league in two-player games. In this experiment, the league consists of the final agent checkpoint of each training run. We compute the Elo for each agent in our single-agent environment by creating simulated two-player games. We reserve a fixed set of environment instances for evaluation. Every agent is evaluated on this evaluation set. We then set up a simulated tournament between every pair of agents. For each pair, we treat each evaluation

Environment	a_B	d_B	a_α	d_α	a_M	d_M	a_{Elo}	d_{Elo}
Starpilot	7.14e−4	0.39	4.08e−3	−0.023	5.3	0.44	−3186	240.97
Craftax	0.28	0.26	1.57e−2	−0.082	0.53	0.43	−1987	223.86

Table 1: Summary of all fitted coefficients and exponents in the experiments presented in Section 4.

environment as an independent competition where the agent achieved a higher return wins. If both agents achieve the same score, it is a tie and counts 0.5 win for each agent. By aggregating over the entire evaluation set, we compute a win rate of one agent against the other. We do this for every pair of agents in the league and obtain a win rate matrix where cell (i, j) denotes the win rate of agent i against agent j. By fitting a Bradley-Terry model (Bradley & Terry, 1952) to this win rate matrix, we can obtain a Elo score for each agent. Note that Elo is a relative metric and thus the solution is not unique. To anchor it, we subtract every agent's Elo by the Elo of the random policy post hoc.

For the performance scaling laws, we used the same data from the allocation experiment. We solved the parabola to find the optimal performance for each training compute C and fitted a scaling law in Elo. Since the Elo score itself is already in log-scale, instead of a power law, we fitted a log-linear law in the form below where d_{Elo} and a_{Elo} are parameters fitted from data:

$$\text{Elo}^*(C) = d_{\text{Elo}} \log_{10} C + a_{\text{Elo}} \tag{4}$$

4 Experiment Results

In this section we present experiment results in Starpilot (Cobbe et al., 2020) and Craftax (Matthews et al., 2024). In the following sections we present the fitted scaling laws for hyperparameters, compute-optimal model sizes, and optimal performance in details. Table 1 summarizes the coefficients and exponents of all fitted laws.

4.1 Hyperparameters

In this experiment, we selected 7 training budgets between 1e14 FLOPs and 1e17 FLOPs. For each budget, we conducted a grid search of 7 different learning rates and 7 different batch sizes with 4 models in different sizes. We used the bootstrapping method described in 3 to identify a set of 1000 competitive batch sizes and learning rates at each training budget. Then we fitted power laws for the learning rate and the batch size using data between 1e14 FLOPs and 1e16 FLOPs and evaluated their predictions at 1e17 FLOPs. In Craftax, 1e14 FLOPs was excluded from the power law fitting because we observed a violation of monotonic model size scaling from 1e14 to 3e14, which we provide a deeper analysis in Section 4.4.

Figure 1 visualizes the distributions of the competitive hyperparameters and the fitted power laws. In both Starpilot and Craftax, the optimal batch size increases as a function of the training FLOPs. The optimal learning rate roughly remains constant with a small negative exponent in Starpilot and a small positive exponent in Craftax. Note that the learning rate scaling trend contradicts the common wisdom that it should decrease as training compute increases. Nonetheless we will see that empirically its predictions are accurate. The heatmaps in Figure 2 visualize where the predicted batch sizes and learning rates locate in the performance contour at 1e17 training FLOPs. In both environments, the predictions locate near the center of the high performance region. In Starpilot the optimal configuration yielded a mean episode return of 32.3 while the predicted configuration yielded 32.02. In Craftax, the optimal mean episode return is 64.76 while the predicted configuration yielded 63.46. Overall, the predicted hyperparameters yield performance close to the emprical optimum, validating the prediction fidelity of the scaling laws.

For the next isoFLOPs profile experiment, we used the scaling laws fitted on all data between 1e14 FLOPs (3e14 for Craftax) and 1e17 FLOPs. The parameters of the fitted laws are shown in Table 1.

4.2 Allocation

We used the isoFLOPs profile approach (Hoffmann et al., 2022) to answer the allocation question. We selected 6 training FLOPs budgets: 1e15, 3e15, 1e16, 3e16, 1e17, and 1e18. For each training

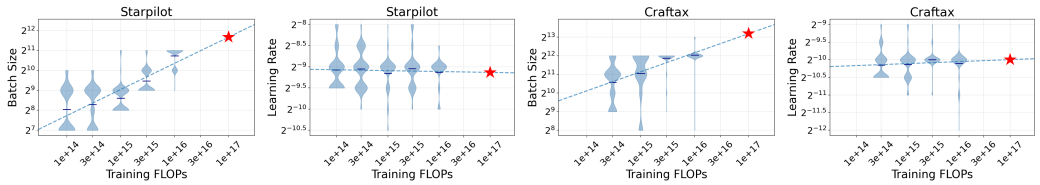

Figure 1: The violin plots show the distributions of the competitive batch sizes and learning rates respectively. The red lines visualize the fitted power laws.

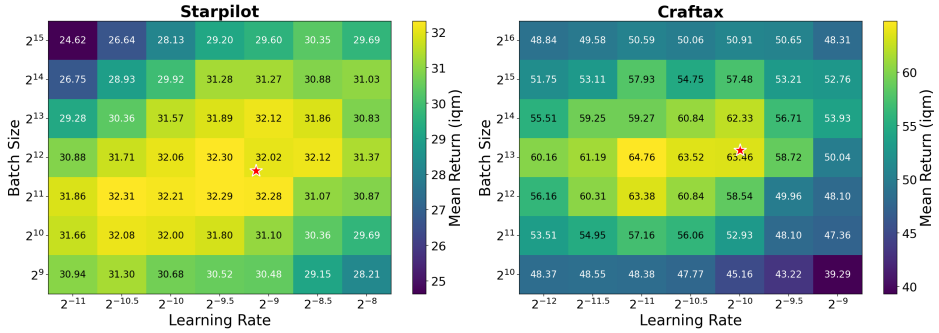

Figure 2: Visualization of performance contour of different hyperparameter configurations at 1e17 training FLOPs. The number in each cell is the mean episode return on the evaluation set, aggregated by IQM across 5 seeds. The red stars denote the predicted configurations by the scaling laws.

budget of C FLOPs, we trained multiple difference model sizes. We kept the depth of the neural network fixed and varied the width to scale the model size. The batch sizes and the learning rates were predicted by the scaling laws fitted in the previous section. To obtain robust results, we ran 10 independent jobs with different random seeds for each configuration. We kept the final agent of each training job and formed a league of them together with a uniform random policy to compute an Elo score for each agent, using the method described in Section 3. For each configuration, we aggregated the Elo scores across all random seeds by IQM. We fitted a parabola to the relationship between $\log_{10}(M)$ and the Elo score by weighted least squares where each data point is weighted by the inverse of its variance across random seeds. Then we extracted its optima to find the corresponding optimal FPF and optimal performance.

The left column in Figure 3 shows the data points and the fitted parabolas and their optima. We fitted a power law in the form of Eq. 3 using data between 1e15 and 1e17 FLOPs and evaluated its prediction at 1e18 FLOPs. The right column in Figure 3 visualizes the fitted laws. The coefficients and exponents of the fitted laws can be found in Table 1. The predicted and the empirically optimal FPFs are summarized in Table 2. In Starpilot, the power law fits the data well. Although the prediction over estimates the optimal FPF by 31%, the performance of the predicted model size is close to optimal (see the yellow circles and stars at 1e18 FLOPs in Figure 4). In Craftax, the fit is less clean. But nevertheless, the FPF prediction is extremely accurate with a relative error of only 3%. Overall the fitted scaling laws demonstrate strong prediction accuracy in both environments, rendering them reliable for informing the model size decisions for large scale training runs.

4.3 Performance

We used the same data from the previous section to fit a law between C and the Elo score in the form of Eq. 4. The parameters of the fitted laws are in Table 1. The left column in Figure 4 visualizes the fitted laws. In both Starpilot and Craftax, the laws fit the data in $[1e15, 1e17]$ well and predict the Elo at 1e18 accurately, resulting in a $+5.84\%$ and -6.37% error in win rates respectively.

We also compare Elo to mean episode return as the performance metric. We fitted similar power laws for the mean episode return, which are shown in the right column of Figure 4. In Starpilot, the mean episode return law fits the data well but extrapolates poorly in comparison to the Elo law.

Environment	predicted FPF	optimal FPF	relative error	Elo of predicted FPF	optimal Elo	win rate error
Starpilot	4.85e8	3.69e8	$+31\%$	1102.6	1110.7	-1.17%
Craftax	2.73e7	2.65e7	$+3\%$	2062.8	2087.5	-3.55%

Table 2: Data for evaluating the prediction accuracy of the optimal model size. We compare the predicted FPF and its realized Elo to the empirical optimum.

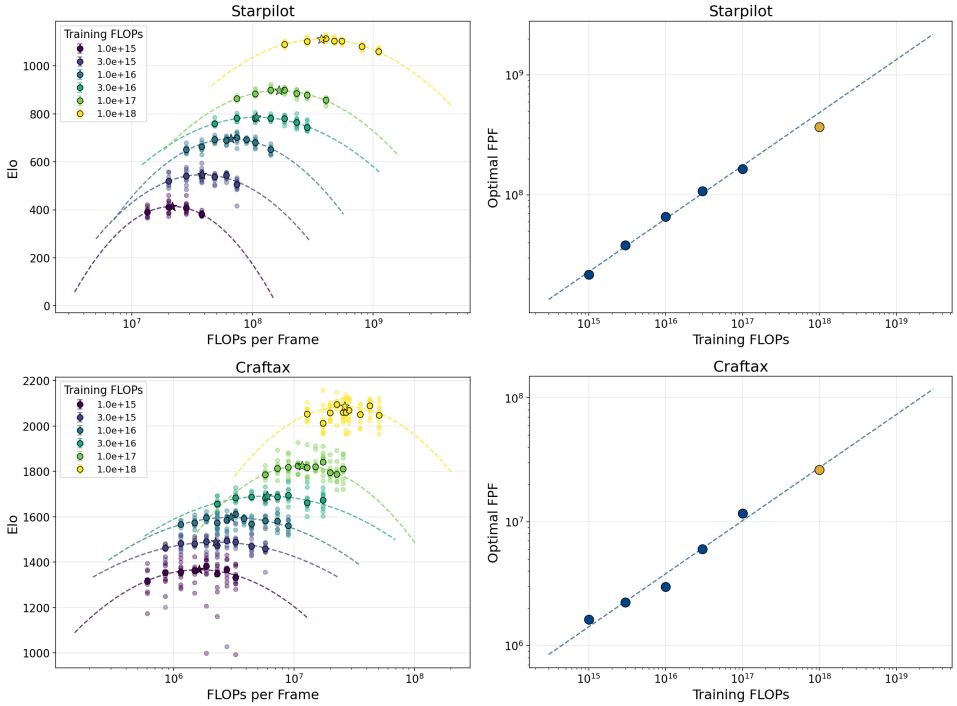

Figure 3: Visualization of data from the isoFLOPs profile experiment in 4.2. The left column visualizes the final performance of each training job and the fitted parabolas. The x-axis denotes the model size measured in FPF. The y-axis denotes the performance measured in Elo. Each parabola corresponds to one training FLOPs. The right column visualizes the derived optimal FPF and the fitted power law. The x-axis is the training FLOPs and the y-axis is the optimal FPF for each training budget. The blue dots are used for fitting the power laws and the yellow dots are used for evaluation.

In Craftax, the mean episode return law does not even fit the data well and the prediction error at 1e18 FLOPs is huge. In both environments, Elo extrapolates significantly better than mean episode return, proving itself as the better performance metric for fitting predictive scaling laws.

In each environment, we also conducted a hero run which trained a model with the size predicted by the FPF scaling law for 1e19 FLOPs. The performance results are summarized in Table 3 and Figure 4. We compare the predicted Elo to the Elo actually achieved. The Elo laws accurately predicts the Elo of larger-scale training jobs for two orders of magnitude more FLOPs. The deviations are $+80.2$ and $+114.8$ for Starpilot and Craftax respecitively, which translate to $+11.3\%$ and $+15.9\%$ deviation in the win rate. We want to highlight the result in Craftax. Through a compute-optimal allocation of training resources, we achieve a new state-of-the-art performance of 52.1%, more than doubling the previous best published baseline of 20.6% (Hamon, 2024). This highlights how scaling laws can be successfully applied to large-scale RL.

4.4 Inverse scaling analysis

In Craftax, we observed a double-crossover between 3e13 and 1e15 FLOPs, demonstrating a local inverse scaling pattern where a smaller model outperforms a larger model as the training budget

Environment	predicted @1e18	optimal @1e18	win rate error @1e18	predicted @1e19	actual @1e19	win rate error @1e19
Starpilot	1151.5	1110.7	+5.84%	1392.5	1312.3	+11.3%
Craftax	2043.0	2087.5	−6.37%	2266.8	2152.0	+15.9%

Table 3: Data for evaluating the accuracy of the Elo predictions. At 1e18 FLOPs, we compare the predicted Elo to the empirically optimal Elo. At 1e19 FLOPs, we compare the predicted Elo to the Elo realized by the predicted optimal configuration.

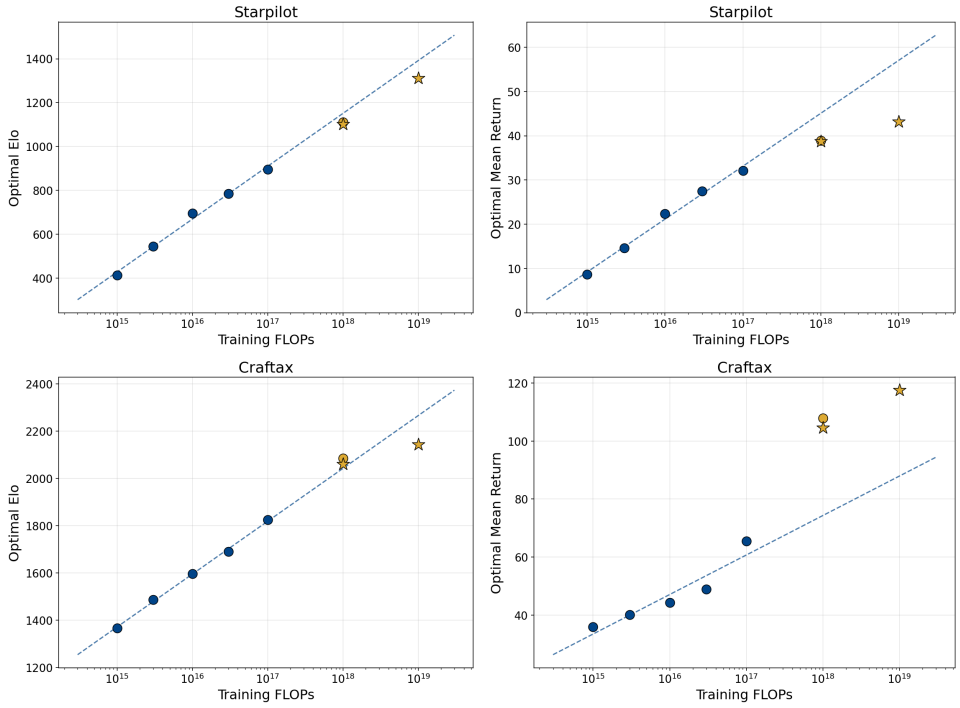

Figure 4: Comparison of different performance metrics. The left column is Elo and the right column is mean episode return. Circles denote empirical optimum. Stars denote the performance of the configuration predicted by the scaling laws. Blue circles are used for fitting the scaling laws. Yellow circles and stars are used for evaluation.

increases. The left plot in Figure 5 shows this phenomenon. The bigger model outperforms the smaller model at 1e14 FLOPs but underperforms at 3e14 FLOPs. Eventually at 1e15 the bigger model performs better again. This observation contradicts the common belief that the optimal model size grows monotonically as we scale up training compute.

We conjecture that this phenomenon stems from a specific design of the game dynamics and the pathological learning dynamics it incurs. In Craftax, the agent must defeat at least 8 enemies on each level to unlock the door to the next level. This rule applies to every level except the first, on which the door to the second level (the "dungeon") is always open since the beginning of the episode, rendering battling optional. However, defeating enemies is relatively easier than entering the dungeon because the door is always located at the center of a "mountain", surrounded by stones. Consequently, the agent learns to defeat enemies before switching to entering the dungeon. During the transition, it stops defeating enemies because it is unnecessary though rewarding. In other words, as the experience accumulates, the agent learns to master the more rewarding skill of entering the dungeon while forgetting the less rewarding skill of battling. This learning dynamics differs drastically from the supervised learning setting where the model needs to fit a growing set of data, which naturally demands more capacity. In this case, however, the agent can reallocate the model capacity from battling to path finding without growing the model size. The smaller model then has an advantage over the larger model because it sees more data under the same compute budget.

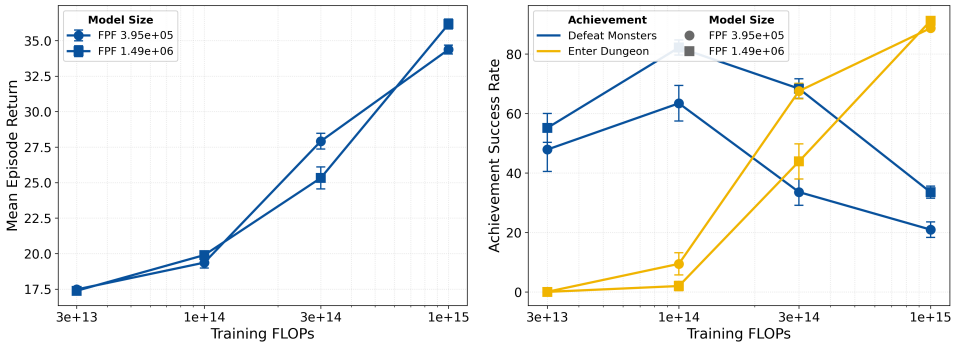

Figure 5: Visualization of the inverse scaling pattern in Craftax. The left plot shows the best performance of two different model sizes across different training FLOPs. The right plot shows the success rate of the enemy defeat achievements and the entering dungeon achievement. Error bars show the standard error across 5 random seeds.

Our conjecture is supported by the achievements the agent unlocked during training, shown in the right plot of Figure 5. Up to 1e14 FLOPs, the agent had little success in entering the dungeon regardless of its model size. Progress mainly comes from better battling skills. However, at 3e14 FLOPs, the agent learned to enter the dungeon while avoiding battles at the same time. Instead of growing its skill set, which demands more model capacity, the agent switches from one skill to another, giving the smaller model advantages.

This pathology of one reward source suppressing the learning signal from another reward source during the early phase of training resembles the Ray interference (Schaul et al., 2019). It is a product of the interaction between the environment dynamics and the learning dynamics. Therefore, a different learning algorithm than actor-critic may demonstrate a different scaling behavior. We leave the comparison of the scaling properties of different algorithms to future work.

5 Conclusion

In this work, we presented a systematic empirical study of scaling laws in RL, investigating hyperparameter configuration, compute allocation, and performance prediction across the Starpilot and Craftax environments. We established that optimal hyperparameters follow power-law relationships with respect to total training compute. Furthermore, we also characterized the optimal allocation between model size and data size as functions of training FLOPs. To address the non-smoothness of mean episode return, we employed a single-agent variant of the Elo score. We demonstrated that the Elo score scales predictably, enabling robust performance extrapolation up to two orders of magnitude in training compute. Guided by these scaling laws, we scaled up a simple actor-critic algorithm to 1e19 training FLOPs and achieved a new state-of-the-art performance of 52.0% in the challenging Craftax benchmark. However, our analysis also uncovered a local inverse scaling pathology in Craftax, which we connected to a pathology known as the Ray interference, illustrating that global scaling trends can obscure non-monotonic local dynamics.

Despite these contributions, several limitations remain to be addressed in future work. First, our empirical study was restricted to a single model-free actor-critic algorithm (A2C); further research is required to verify if our findings generalize to other RL algorithms such as value-based or model-based algorithms. Second, while the Elo score offers predictable mathematical scaling, it is fundamentally a relative metric within a closed league of evaluated agents, lacking direct semantic grounding to the underlying task. Consequently, in domains where rewards carry critical semantic meanings, Elo-based scaling laws may be less effective at providing informative or absolute performance predictions. Finally, our discovery of Ray interference in Craftax indicates that RL scaling is highly sensitive to the specific alignment of environmental dynamics and reward structures, suggesting that a single, universal scaling law may not apply across all arbitrary RL environments. Designing algorithms resilient to Ray interference remains a critical avenue for future research.

AI USE STATEMENT

In this work, we used generative AI tools for [**implement methods, design or provide feedback on research methodology or experiments, clean and reformat dataset, interpret results**]. We have not used generative AI tools for [generate synthetic data sets, help develop theoretical models or conceptual framework, propose or refine hypotheses] and [formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, assist with translation, support qualitative and thematic data analysis.] are not applicable to this work. Additionally, we used generative AI tools for [**edit a research paper to improve readability**]. We have reviewed all AI-assisted work. We reviewed the AI-generated code for experiment and data analysis. We proofread the text revised by AI tools. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafal JĂłzefowicz, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, et al. Deepseek llm: Scaling open-source language models with longtermism. *arXiv preprint arXiv:2401.02954*, 2024.

Ralph Allan Bradley and Milton E. Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

Zhoujun Cheng, Yutao Xie, Yuxiao Qu, Amrith Setlur, Shibo Hao, Varad Pimpalkhute, Tongtong Liang, Feng Yao, Zhengzhong Liu, Eric Xing, et al. Isocompute playbook: Optimally scaling sampling compute for llm rl. *arXiv preprint arXiv:2603.12151*, 2026.

Karl Cobbe, Chris Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *International conference on machine learning*, pp. 2048–2056. PMLR, 2020.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

Arpad E. Elo. *The Rating of Chessplayers, Past and Present*. Arco Pub., New York, 1978. ISBN 0668047216.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International conference on machine learning*, pp. 1407–1416. PMLR, 2018.

Preston Fu, Oleh Rybkin, Zhiyuan Paul Zhou, Michal Nauman, Pieter Abbeel, Sergey Levine, and Aviral Kumar. Compute-optimal scaling for value-based deep rl. *Advances in Neural Information Processing Systems*, 38:160363–160402, 2026.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Gautier Hamon. transformerXL_PPO_JAX, July 2024. URL `https://inria.hal.science/hal-04659863`.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv: Learning*, 2016. URL https://arxiv.org/abs/1606.08415.

Tom Henighan, Jared Kaplan, Mor Katz, Mark Chen, Christopher Hesse, Jacob Jackson, Heewoo Jun, Tom B Brown, Prafulla Dhariwal, Scott Gray, et al. Scaling laws for autoregressive generative modeling. *arXiv preprint arXiv:2010.14701*, 2020.

Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md Mostofa Ali Patwary, Yang Yang, and Yanqi Zhou. Deep learning scaling is predictable, empirically. *arXiv preprint arXiv:1712.00409*, 2017.

Jacob Hilton, Jie Tang, and John Schulman. Scaling laws for single-agent reinforcement learning. *arXiv preprint arXiv:2301.13442*, 2023.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Andy L Jones. Scaling scaling laws with board games. *arXiv preprint arXiv:2104.03113*, 2021.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Devvrit Khatri, Lovish Madaan, Rishabh Tiwari, Rachit Bansal, Venkata Sai Surya Subramanyam Duvvuri, Manzil Zaheer, Inderjit Dhillon, David Brandfonbrener, and Rishabh Agarwal. The art of scaling reinforcement learning compute for llms. In *International Conference on Learning Representations*, volume 2026, pp. 72438–72467, 2026.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun (eds.), *ICLR (Poster)*, 2015. URL http://dblp.uni-trier.de/db/conf/iclr/iclr2015.html#KingmaB14.

Michael Matthews, Michael Beukman, Benjamin Ellis, Mikayel Samvelyan, Matthew Thomas Jackson, Samuel Coward, and Jakob Nicolaus Foerster. Craftax: A lightning-fast benchmark for open-ended reinforcement learning. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 35104–35137. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/matthews24a.html.

Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In Maria Florina Balcan and Kilian Q. Weinberger (eds.), *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 1928–1937, New York, New York, USA, 20–22 Jun 2016. PMLR. URL https://proceedings.mlr.press/v48/mniha16.html.

Niklas Muennighoff, Alexander Rush, Boaz Barak, Teven Le Scao, Nouamane Tazi, Aleksandra Piktus, Sampo Pyysalo, Thomas Wolf, and Colin A Raffel. Scaling data-constrained language models. *Advances in Neural Information Processing Systems*, 36:50358–50376, 2023.

Oren Neumann and Claudius Gros. Scaling laws for a multi-agent reinforcement learning model. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=ZrEbzL9eQ3W.

Jonathan S. Rosenfeld, Amir Rosenfeld, Yonatan Belinkov, and Nir Shavit. A constructive prediction of the generalization error across scales. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=ryenvpEKDr.

Tom Schaul, Diana Borsa, Joseph Modayil, and Razvan Pascanu. Ray interference: a source of plateaus in deep reinforcement learning. *arXiv preprint arXiv:1904.11455*, 2019.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

Zelin Tan, Hejia Geng, Xiaohang Yu, Mulei Zhang, Guancheng Wan, Yifan Zhou, Qiang He, Xiangyuan Xue, Heng Zhou, Yutao Fan, et al. Scaling behaviors of llm reinforcement learning post-training: An empirical study in mathematical reasoning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 31300–31319, 2026.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *nature*, 575(7782):350–354, 2019.

A Training Algorithm Details

In both Starpilot and Craftax, we train agents using the A2C (Mnih et al., 2016) algorithm.

A.1 Neural Network Architectures.

In Starpilot experiments, we use the IMPALA (Espeholt et al., 2018) residual convolutional neural network architecture (He et al., 2016) with a shared torso between the policy and the value function. Layer normalization (Ba et al., 2016) is applied between every layer and its following activation function. We vary the model size by the number convolutional channels of the first layer. The width of subsequent layers scale proportionally.

In Craftax we use the symbolic observations. We apply a 1×1 convolutional layer with E channels to the map grid. We then flatten the grid and apply a linear layer with dimension $H = 16E$ to compress the map to a compact vector representation. The inventory is encoded to a vector of the same dimensionality H by a linear layer. We fuse the map and the inventory by adding their vectors together. Note that we apply layer normalization to their vectors independently *before* fusion. The fused vector is fed into a LSTM (Hochreiter & Schmidhuber, 1997) cell with H hidden units to handle partial observability. Lastly, we apply linear layers to the output of the LSTM cell to map the state representation to the policy logits and the value function. We used GELU (Hendrycks & Gimpel, 2016) activation. Similarly, we apply layer normalization before every activation layer. We control the model size by varying the number of channels in the 1×1 convolutional layer E.

A.2 Hyperparameters

In Starpilot experiments, we use a discount factor of 0.999. We set the rollout length to a constant 16. The weight for the value function loss is 0.5 and the entropy regualrization strength is 0.01. We use the Adam (Kingma & Ba, 2015) optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 1\text{e}{-8}$. The learning rate is linearly decayed to 0 through training. We clip the gradient by global norm to 0.5.

In Craftax, we use a discount factor of 0.999. We set the rollout length to a constant 16. We break each batch of rollouts into 4 minibatches and thus apply 4 gradient updates per batch. We do minibatching to improve the overall throughput of the training system because we can employ more parallel environments. The batch size in Section 4 is the number of observations in each minibatch. We apply VTrace (Espeholt et al., 2018) to handle off-policyness caused by minibatch.

The weight for the value function loss is 0.5 and the entropy regualrization strength is 0.001. We use the Adam (Kingma & Ba, 2015) optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.99$, and $\epsilon = 1\text{e}{-8}$. The learning rate is linearly decayed to 0 through training. We clip the gradient by global norm to 0.5.

B Experiment Configurations

Table 4 and Table 5 summarize the configuration of the grid search in the hyperparameter scaling law experiments for Starpilot and Craftax respectively. Table 6 and Table 7 summarize the configuration for the isoFLOPs profile experiments in Starpilot and Craftax respectively.

FLOPs	Model width	Batch size	Learning rate
1e14	$\{1, 2, 3, 4\}$	$\{2^6, 2^7, 2^8, 2^9, 2^{10}, 2^{11}, 2^{12}\}$	$\{2^{-11}, 2^{-10.5}, 2^{-10}, 2^{-9.5}, 2^{-9}, 2^{-8.5}, 2^{-8}\}$
3e14	$\{2, 3, 4, 6\}$	$\{2^6, 2^7, 2^8, 2^9, 2^{10}, 2^{11}, 2^{12}\}$	$\{2^{-11}, 2^{-10.5}, 2^{-10}, 2^{-9.5}, 2^{-9}, 2^{-8.5}, 2^{-8}\}$
1e15	$\{3, 4, 6, 8\}$	$\{2^7, 2^8, 2^9, 2^{10}, 2^{11}, 2^{12}, 2^{13}\}$	$\{2^{-11}, 2^{-10.5}, 2^{-10}, 2^{-9.5}, 2^{-9}, 2^{-8.5}, 2^{-8}\}$
3e15	$\{4, 6, 8, 12\}$	$\{2^7, 2^8, 2^9, 2^{10}, 2^{11}, 2^{12}, 2^{13}\}$	$\{2^{-11}, 2^{-10.5}, 2^{-10}, 2^{-9.5}, 2^{-9}, 2^{-8.5}, 2^{-8}\}$
1e16	$\{6, 8, 12, 16\}$	$\{2^7, 2^8, 2^9, 2^{10}, 2^{11}, 2^{12}, 2^{13}\}$	$\{2^{-11}, 2^{-10.5}, 2^{-10}, 2^{-9.5}, 2^{-9}, 2^{-8.5}, 2^{-8}\}$
3e16	$\{8, 12, 16, 24\}$	$\{2^8, 2^9, 2^{10}, 2^{11}, 2^{12}, 2^{13}, 2^{14}\}$	$\{2^{-11}, 2^{-10.5}, 2^{-10}, 2^{-9.5}, 2^{-9}, 2^{-8.5}, 2^{-8}\}$
1e17	$\{8, 12, 16, 24\}$	$\{2^9, 2^{10}, 2^{11}, 2^{12}, 2^{13}, 2^{14}, 2^{15}\}$	$\{2^{-11}, 2^{-10.5}, 2^{-10}, 2^{-9.5}, 2^{-9}, 2^{-8.5}, 2^{-8}\}$

Table 4: Grid search configurations for fitting hyperparameter scaling laws in Starpilot.

FLOPs	Model width	Batch size	Learning rate
3e14	$\{3, 4, 6, 8\}$	$\{2^7, 2^8, 2^9, 2^{10}, 2^{11}, 2^{12}, 2^{13}\}$	$\{2^{-12}, 2^{-11.5}, 2^{-11}, 2^{-10.5}, 2^{-10}, 2^{-9.5}, 2^{-9}\}$
1e15	$\{4, 6, 8, 12\}$	$\{2^8, 2^9, 2^{10}, 2^{11}, 2^{12}, 2^{13}, 2^{14}\}$	$\{2^{-12}, 2^{-11.5}, 2^{-11}, 2^{-10.5}, 2^{-10}, 2^{-9.5}, 2^{-9}\}$
3e15	$\{6, 8, 12, 16\}$	$\{2^8, 2^9, 2^{10}, 2^{11}, 2^{12}, 2^{13}, 2^{14}\}$	$\{2^{-12}, 2^{-11.5}, 2^{-11}, 2^{-10.5}, 2^{-10}, 2^{-9.5}, 2^{-9}\}$
1e16	$\{6, 8, 12, 16\}$	$\{2^9, 2^{10}, 2^{11}, 2^{12}, 2^{13}, 2^{14}, 2^{15}\}$	$\{2^{-12}, 2^{-11.5}, 2^{-11}, 2^{-10.5}, 2^{-10}, 2^{-9.5}, 2^{-9}\}$
3e16	$\{8, 12, 16, 24\}$	$\{2^9, 2^{10}, 2^{11}, 2^{12}, 2^{13}, 2^{14}, 2^{15}\}$	$\{2^{-12}, 2^{-11.5}, 2^{-11}, 2^{-10.5}, 2^{-10}, 2^{-9.5}, 2^{-9}\}$
1e17	$\{12, 16, 24, 32\}$	$\{2^{10}, 2^{11}, 2^{12}, 2^{13}, 2^{14}, 2^{15}, 2^{16}\}$	$\{2^{-12}, 2^{-11.5}, 2^{-11}, 2^{-10.5}, 2^{-10}, 2^{-9.5}, 2^{-9}\}$

Table 5: Grid search configurations for fitting hyperparameter scaling laws in Craftax.

FLOPs	Model widths	Batch size	Learning rate
1e15	$\{4, 5, 6, 7\}$	2^9	$2^{-9.12}$
3e15	$\{5, 6, 7, 8, 9, 10\}$	2^{10}	$2^{-8.97}$
1e16	$\{6, 7, 8, 9, 10, 11, 12, 14\}$	2^{10}	$2^{-9.35}$
3e16	$\{8, 10, 12, 14, 16, 18, 20\}$	2^{11}	$2^{-9.19}$
1e17	$\{10, 12, 14, 16, 18, 20, 24\}$	2^{12}	$2^{-9.07}$
1e18	$\{16, 20, 24, 26, 28, 34, 40\}$	2^{13}	$2^{-9.30}$
1e19	$\{43\}$	2^{14}	$2^{-9.53}$

Table 6: Configurations for the IsoFLOPs profile experiment in Starpilot.

FLOPs	Model widths	Batch size	Learning rate
1e15	$\{5, 6, 7, 8, 9, 10, 11, 12\}$	2^{11}	$2^{-10.17}$
3e15	$\{6, 7, 8, 9, 10, 11, 12, 14, 16\}$	2^{11}	$2^{-10.51}$
1e16	$\{7, 8, 9, 10, 11, 12, 13, 14, 16, 18, 20\}$	2^{12}	$2^{-10.38}$
3e16	$\{10, 12, 14, 16, 18, 20, 24, 28\}$	2^{12}	$2^{-10.71}$
1e17	$\{16, 18, 20, 22, 24, 26, 28, 30, 32, 34\}$	2^{13}	$2^{-10.58}$
1e18	$\{24, 28, 30, 32, 34, 35, 36, 40, 44, 48\}$	2^{14}	$2^{-10.79}$
1e19	$\{58\}$	2^{15}	$2^{-11.00}$

Table 7: Configurations of the IsoFLOPs profile experiment in Craftax.