Suboptimal Data Can Bottleneck Scaling

Anonymous Authors[1]

Abstract

Deep learning has been shown to reliably improve in performance on supervised learning tasks when scaling up data, compute, and parameters. In this work, we argue that properly understanding the impact of scale requires a nuanced understanding of dataset composition. Towards this end, we design experiments in the domain of offline reinforcement learning to disentangle the effects of data *quantity* and *quality*. Our results comprehensively confirm that performance is bottlenecked by the quality of the data, even in the limit of parameters, compute, and dataset size. Furthermore, we show that the performance of offline reinforcement learning algorithms obeys reliable scaling laws in these settings, allowing performance-at-scale to be extrapolated from a smaller set of experiments.

1. Introduction

In recent years, deep learning has been shown to be a powerful technique, obtaining state-of-the-art performance on a wide variety of benchmarks. Central to the success of deep learning is its ability to improve with scale: increasing the amount of data, parameters, and computation spent on a deep learning algorithm translates into reliable improvements in generalization (Nakkiran et al., 2021; Kaplan et al., 2020). Thanks to Moore's law and related economic factors, the amount of compute available with which to train deep learning models has increased rapidly with each passing year. Large models such as GPT-3 (Brown et al., 2020) and DALL-E 2 (Ramesh et al., 2022) have demonstrated that neural networks can acquire remarkable capabilities simply by scaling up existing methods.

In the deep learning literature, it is common to characterize datasets along a single scalar axis: size. This metric is a simple and effective way to capture an important dimension of how datasets impact performance. In particular, when a dataset is drawn as an IID subset of some underlying data-generating process (as is most often the case for academic machine learning), the law of large numbers ensures that all datasets of the same sufficiently-large size are nearly identical.

However, the size of a dataset is ultimately a coarse description of its contents. It is natural to want to understand the interaction between datasets and learning in richer detail. As a first step towards this goal, we propose to consider a dataset not just in terms of its size, or data *quantity*, but also its data *quality*. Concretely, in this work, we study the question: **what are the limits of learning from low-quality data?** This question is of particular interest thanks to the empirical success of models trained on Internet data (Ramesh et al., 2022; Brown et al., 2020). These models have been shown to have impressive capabilities, such as the ability to generate photorealistic images in a variety of styles (Ramesh et al., 2022), or demonstrate simple mathematical reasoning (Wei et al., 2022). Scaling laws (Kaplan et al., 2020) predict that these capabilities will continue to increase as the models are made ever larger. However, their source of training data, the Internet, can be reasonably considered to be a high-quantity, low-quality dataset. Understanding of the limitations of learning from low-quality data therefore has implications for the capabilities we anticipate these models to eventually acquire.

A difficulty in answering this question is that the quality of a dataset is a less well-defined concept than its size. Quality is task-dependent: the same dataset might contain high-quality data for one task, but low-quality for another. In order to operationalize our motivating question, we turn to the setting of *offline reinforcement learning* (Levine et al., 2020). The goal of offline reinforcement learning, which we formally describe in Section 2.2, is to learn a policy from previously-collected data. The expected return of the learned policy serves as a quantitative measurement of a model's capabilities. In this setting, high-quality data has a natural interpretation as data collected by a high-performing expert.

For reinforcement learning environments, we have complete control over the data generation process. Any dataset can be constructed by simply deploying existing agents in the

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

environment and recording their interactions. We therefore can train models for offline RL in the infinite-data setting, where no datapoint is ever seen by the model more than once. This allows us to assess their asymptotic performance in the limit of compute and data. A complete description of our experimental setup is given in Section 3.

We find that in all cases, asymptotic performance is dependent on the quality of the dataset; no current algorithm is capable of surpassing the best expert by an arbitrary margin. To establish that these results are broadly applicable, we study five different algorithms on nine different environments across two different domains at a variety of model sizes, and see similar trends across all of them. These results are described in detail in Section 4. Furthermore, we fit broken neural scaling laws (Caballero et al., 2022) to these outcomes, and that find performance scales predictably, analogously to supervised learning.

Our central result is a comprehensive empirical confirmation that low-quality data can be a bottleneck to scaling. We also make the following auxiliary contributions: an experimental procedure for evaluating algorithms for offline reinforcement learning which disentangles data quality from other factors; an empirical comparison between the performance-at-scale of current algorithms on several domains; and a demonstration that the performance of offline reinforcement algorithms obeys reliable scaling laws.

2. Background

2.1. Deep Learning

Deep learning (Goodfellow et al., 2016) is a family of algorithms for function approximation, which is defined as the problem setting whose goal is to learn a function $\hat{f} : \mathcal{X} \to \mathcal{Y}$ that most closely resembles some (possibly stochastic) ground-truth function $f : \mathcal{X} \to \mathcal{Y}$, based on a dataset D of input-output pairs $D = \langle (x_0, y_0), (x_1, y_1), ..., (x_N, y_N) \rangle$ where each $x_i \in \mathcal{X}$ and each $y_i = f(x_i)$. In particular, it fits a parameterized function given by a deep neural network (LeCun et al., 1995) by using stochastic gradient descent (SGD) to minimize some objective (Robbins & Monro, 1951; Rumelhart et al., 1986).

2.2. Offline Reinforcement Learning

In offline reinforcement learning (RL), the environment is represented as a Markov Decision Process (MDP), denoted $\mathcal{M} := \langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, \gamma, \rho \rangle$, with state space $\mathcal{S}$, action space $\mathcal{A}$, reward function $\mathcal{R}$, transition function $\mathcal{P}$, discount γ, and start-state distribution ρ. Policies $\pi \in \Pi$ map states to actions. A policy interacting with the environment yields a sequence of states, actions, and rewards known as a trajectory. Starting in any initial state s_0,

$$a_t \sim \pi(s_t) \qquad r_t \sim R(s_t, a_t) \qquad s_{t+1} \sim P(s_t, a_t)$$

The return of a policy π starting from some state s_0, denoted $\mathcal{G}(\pi, s_0)$, is a stochastic function which yields a sample of the discounted sum of rewards along the trajectory,

$$\mathcal{G}(\pi, s_0) = \sum_{t=0}^{\infty} \gamma^t r_t$$

The value of a policy π is defined as $\mathbb{E}_{s \sim \rho}[\mathcal{G}(\pi, s)]$. The goal of offline reinforcement learning is to find the policy with the highest value, given only a dataset D containing a collection of trajectories.

2.3. Scaling Laws

Neural scaling laws are functional forms that model and extrapolate the scaling behaviors of artificial neural networks. A scaling behavior is a closed-form expression relating a performance metric to an attribute of the learning problem, typically the number of model parameters, the amount of compute used for training, or the training dataset size.

Previous work on scaling laws by Kaplan et al. (2020) show that the loss of a language model scales as a power-law with model size, dataset size, and the amount of compute used for training, and that this relationship holds across several orders of magnitudes. The reliability is such that a parameterized power-law model fit to the first several orders of magnitude can extrapolate to larger orders of magnitude, on several dimensions of variation. Subsequent work extended this result to a variety of settings and considers other functional forms (Hoffmann et al., 2022; Aghajanyan et al., 2023; Maloney et al., 2022; Caballero et al., 2022).

Broken Neural Scaling Laws (BNSL) are neural scaling laws that have been shown to accurately model and extrapolate the scaling behavior of deep RL algorithms, in addition to the scaling behavior of many other things that involve artificial neural networks (Caballero et al., 2022). In this work, we use BNSL to model and extrapolate the scaling behavior of offline deep reinforcement learning algorithms.

3. Experimental Design

3.1. Datasets In More Detail

A dataset is an essential ingredient of both supervised learning and offline reinforcement learning. The contents of the dataset directly determine what solution is learned, and performance of any modern algorithm is highly dependent on the contents of the dataset it was trained on. A powerful approach to improving capabilities is collecting more data for training.

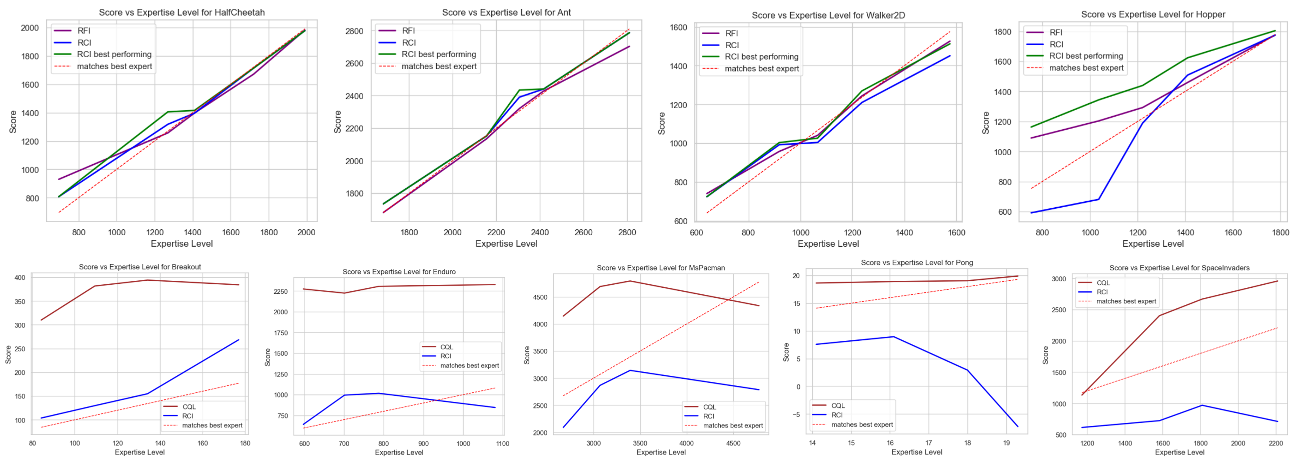

Figure 1. Comparison of performance of algorithms on PyBullet and Atari.

Many previous works study how datasets impact performance. However, most current literature (Kaplan et al., 2020; Maloney et al., 2022; Chen et al., 2021; Kumar et al., 2023) considers only a one-dimensional axis of dataset quality: dataset size, as measured by the number of distinct datapoints in the training set. In general, this coarse measurement is far from the complete picture. For example, duplicating every point in a dataset would double its size, but not improve performance at all; this follows from the fact that the distribution of minibatches sampled from the dataset during training would be unchanged, and thus, the function learned will be identical. In this work, we seek to understand the dataset in more detail.

To this end, we propose a natural generative process for modeling how a dataset might come to be. In our framework, some number of *experts* each provide some number of *datapoints*, which together form the dataset. Each expert yields datapoints which follow a particular distribution, and so the overall distribution from which training points emerge is the mixture of these distributions.

It is natural to see how this model applies to offline reinforcement learning, where experts take the form of policies, and datapoints come from environment interaction. We also believe our model is also a natural framework to understand datasets from supervised learning. For example, Internet-scale datasets (Schuhmann et al., 2022) such as those used to train GPT-3 and DALLE-2 can be understood as coming from a mixture of individual human experts, each choosing which text and images to upload onto the Internet according to their own idiosyncratic distribution.

Under this model, we can characterize datasets using three different axes: *size*, *diversity*, and *quality*. Dataset size measures the number of datapoints available to be seen during training. Dataset diversity measures the number of distinct experts used to generate the datapoints. Finally, dataset quality measures the extent to which the experts which generated the dataset produce valuable data. In the context of offline reinforcement learning, where each expert is a policy and the goal is to learn a high-value policy, we consider a high-quality dataset to be one which contains trajectories from high-value experts.

3.2. Constructing Datasets

The primary goal of our research is to assess the extent to which performance can improve by increasing dataset scale, without increasing its quality or diversity. To this end, we investigate the relationship between scaling compute, parameters, dataset size, and performance, under various levels of dataset quality. In order to construct datasets of systematically varying quality for each environment, our experiments require three steps of setup.

- **Acquire expert policies.** Obtain a diverse set of expert policies, with a wide variety of skill levels. In our experiments, we get these from various checkpoints of an online reinforcement learning algorithm.
- **Collect data.** For each policy, collect a large number of episodes of interaction, storing it in a dataset. Additionally, estimate the value of each policy as the mean return of the collected episodes.
- **Construct datasets of varying expertise.** For each desired expertise level, filter the set of policies to exclude any whose value is greater than a given amount, and further filter it to exclude all but P policies, subsampled evenly.[1] The dataset of that expertise level is

[1]This second filtering step ensures that datasets of all expertise levels come from the same number of distinct experts, ensuring that policy diversity is not a confounding factor for performance. In this work, we fix $P = 100$, and leave measurement of the

defined to be the concatenation of all data collected by those policies which remain.

Our notion of expertise is natural, because many real-world data distributions will come from many experts with varying levels of performance. We hypothesize that any similarly-reasonable approach to defining expertise would yield qualitatively similar results, and hope future work will validate this claim.

3.3. Environment Details

We run experiments in two popular benchmark domains: robotic control in PyBullet physics simulator (Coumans & Bai, 2016–2021), and game play on Atari games in the Arcade Learning Environment (Bellemare et al., 2013).

3.3.1. PyBullet

We evaluate four PyBullet environments: HalfCheetah, Hopper, Walker2D, and Ant. In order to simplify implementation, we discretize the action space using a bang-bang control scheme (Bellman et al., 1956; Seyde et al., 2021). For an environment whose action space is of dimension $|D|$, this yields $2^{|D|}$ discretized actions. To add stochasticity, we sample each dimension of the selected action from a uniform distribution between either $(0.8, 1)$ or $(-1, -0.8)$.

To acquire our set of diverse expert policies, we run Stable-Baselines3 PPO (Raffin et al., 2019) for 2,000,000 steps, checkpointing every 20,000 steps. We repeat this for five seeds, yielding 500 policies overall.

3.3.2. Atari

We evaluate on five commonly-studied Atari 2600 games: Breakout, Enduro, Ms Pacman, Pong, and Space Invaders. We use the DQN replay dataset released by Agarwal et al. (2020). This dataset was collected by running 5 independent runs of DQN (Mnih et al., 2015) on each game, and recording all interactions for each run. We use the first 40 million environment steps in each of the runs and divide each of these runs into 40 chunks, for 200 total chunks, and treat each chunk as having been collected by a single stationary policy.[2] For efficiency reasons, we use the tfds version of the dataset (Gulcehre et al., 2020) rather than generating this ourselves.

impact of this factor as a topic for future work.

[2]In reality, due to how DQN is implemented, each chunk contains episodes from a constantly-evolving nonstationary policy. Making this assumption saved significant computational expense, and we do not believe it impacted our results. This assumption is further justified by the fact that, for data collected by any non-stationary policy, there is a stationary policy with the same data distribution (Laroche et al., 2022).

3.4. Algorithms

We study three well-known families of offline reinforcement algorithms: imitation learning, conditional imitation learning, and dynamic programming. We select a well-known algorithm from each family to test.

3.4.1. Imitation

A simple baseline for offline reinforcement learning is imitation learning (Hussein et al., 2017), which reduces the problem of decision-making to a simple classification task. Each state is labeled with the action taken by an expert in that state. As is standard in supervised learning, we minimize the negative conditional log-likelihood of the labels. A well-trained model will choose actions according to the same policy as the data-generating experts.

When a dataset contains trajectories from policies of varying ability (as ours do), an agent trained to imitate policies in the dataset will sometimes take bad actions. A straightforward way to mitigate this limitation is to filter out bad actions, and only learn to imitate good actions. We refer to this family of methods as *filtered imitation*. A standard way to do this is filter out actions which do not belong to high-return trajectories, which we call **return-filtered imitation (RFI)**. This algorithm is commonly used as a baseline for offline reinforcement learning (Chen et al., 2021; Brandfonbrener et al., 2022). In our experiments, we filter out trajectories with returns below the top 10%.

3.4.2. Conditional Imitation

Another approach to offline reinforcement learning, which we refer to as *conditional imitation*, is to train an imitation-learning agent which is additionally conditioned on some goal or outcome. This family of algorithms has recently grown in popularity, and has also been studied as upside-down RL (Štrupl et al., 2022), decision transformer (Chen et al., 2021), and return-conditioned supervised learning (Brandfonbrener et al., 2022).

An important algorithmic decision in conditional imitation is deciding what to condition on. In offline reinforcement learning, a common choice is to condition on return, an algorithm which we study in this work as **return-conditional imitation (RCI)**. While a variety of conditioning schemes are possible (Brandfonbrener et al., 2022), in this work, we study the simplest approach: condition each state within a trajectory on the sum of rewards across the entire trajectory.

A well-trained return-conditional imitation agent can embody a variety of policies when prompted with the correct context. For example, an agent trained via return-conditional imitation can be prompted to act in a way that acquires either high or low returns. Crucially, such agents could in principle achieve arbitrarily good performance by

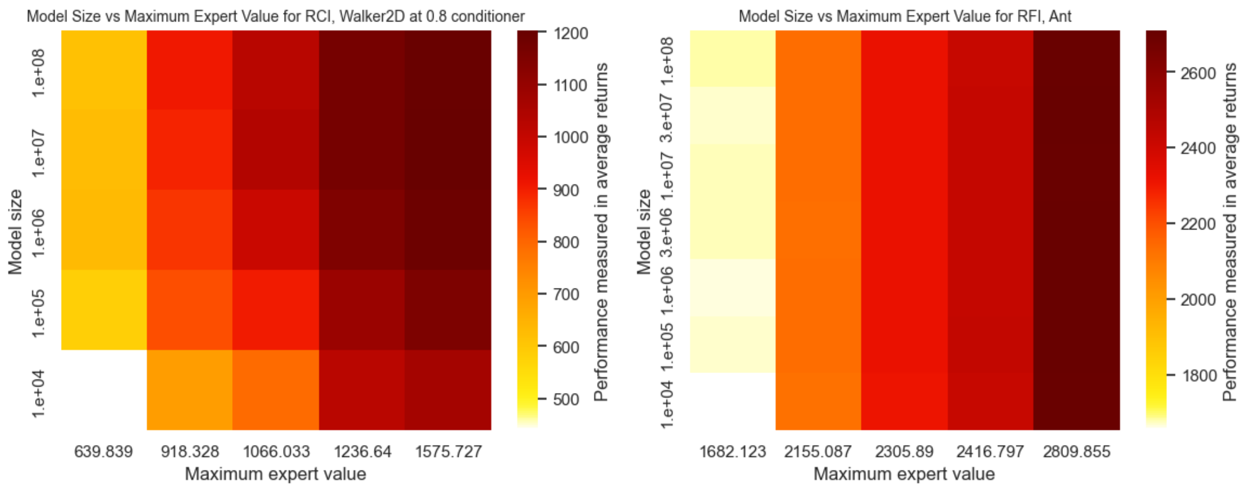

Figure 2. Tradeoff between model size and dataset quality

simply conditioning on an arbitrarily high return. Of course, this requires generalization beyond the data distribution, which may be challenging. An important open question is to characterize the conditions and extent to which this generalization is possible.

In this work, we always choose our conditioner based on the returns in the training data. In reporting our results, we describe our conditioners using real numbers, e.g. $c = 1.0$. This is translated into the actual value used to condition the model according to $\mathcal{G}_{\min} + c(\mathcal{G}_{\max} - \mathcal{G}_{\min})$, where $\mathcal{G}_{\min}, \mathcal{G}_{\max}$ correspond to the minimum and maximum returns of any trajectory present in the training data.

3.4.3. Dynamic Programming

The final family of algorithms for offline reinforcement learning that we consider is dynamic programming (Sutton & Barto, 2018). These algorithms estimate the value of each state, and iteratively apply the Bellman optimality operator, approaching an optimal fixed point. A typical implementation, inspired by online deep reinforcement learning algorithms like DQN (Mnih et al., 2015) or Rainbow (Hessel et al., 2018), parameterizes a Q-value function using a neural network and minimizes the expected Bellman error on transitions sampled from the dataset.

In the offline RL setting, the fixed point reached by a dynamic programming algorithm has poor guarantees, unless it incorporates pessimism or a policy constraint (Buckman et al., 2020). This is reflected in the practical performance of these algorithms as well (Fu et al., 2020; Gulcehre et al., 2020). Therefore, we study **Conservative Q-Learning (CQL)** (Kumar et al., 2020), a dynamic programming algorithm that incorporates pessimism and achieves state-of-the-art performance (Kumar et al., 2023). CQL implements pessimism by adding an additional regularizing term to the loss, which penalizes the value function for predicting high values on actions that do not appear in the dataset. The degree of pessimism is controlled by a hyperparameter $\alpha \geq 0$, where $\alpha = 0$ corresponds to the unpenalized objective.

4. Results

For each of our environments, we constructed five sets of experts as described in Section 3, corresponding to five expertise levels. For each set of experts, we trained models on an infinite stream of expert interactions until convergence, at a variety of model sizes.

Our first domain is PyBullet (Coumans & Bai, 2016–2021), where we study two algorithms: RFI, and RCI with various conditioners. Both algorithms use a simple feedforward network architecture; the RCI architecture has extra input dimensions to represent the return, and uses a square-wave embedding function to encode the scalar return into a 32-dimensional vector. We scale the network to have parameter counts between 10K and 300M, adjusting the network width to maintain an aspect ratio of 256. We evaluate data expertise values in $\{0.2, 0.4, 0.6, 0.8, 1\}$. Full implementation details for the PyBullet experiments can be found in our open-source repository.[3]

Our second domain is the ALE (Bellemare et al., 2013). We study two algorithms in this domain, RCI and CQL. Based on the results of Schmidt & Schmied (2021); Agarwal et al. (2022), we base our offline agents on top of distributional Rainbow (Hessel et al., 2018) combined with Impala-CNN, which corresponds to a 15-layer ResNet. To vary model capacity, we scale the feature dimension of every layer by

[3]`github.com/anonymous`

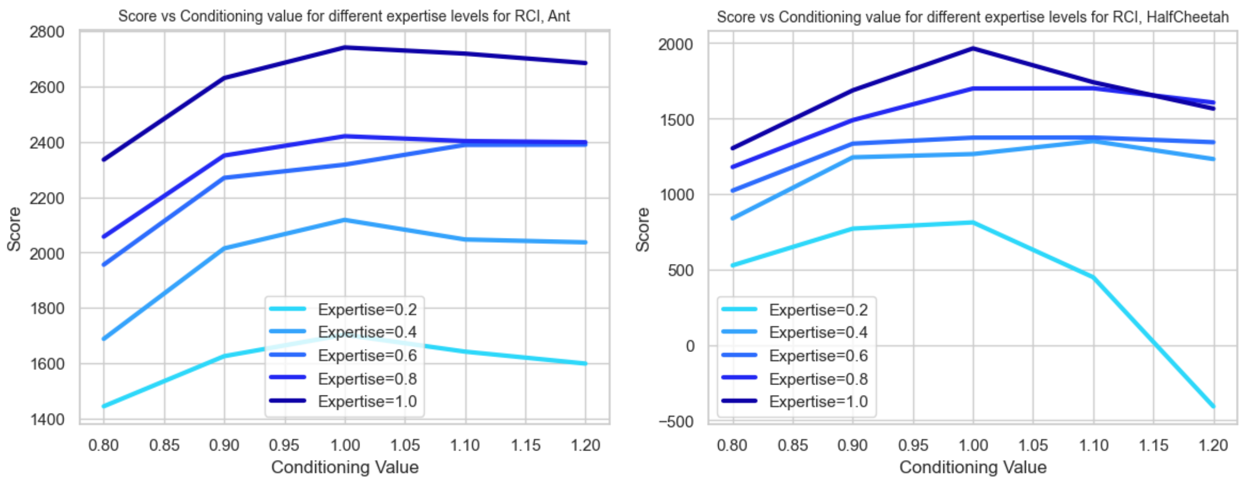

Figure 3. Ablatinon study for different conditioning values for RCI for HalfCheetah and Ant.

factors of $\{0.5, 1, 2, 3, 4, 6\}$, resulting in parameter counts ranging from 600K to 75.8 M. We evaluate data expertise values in $\{0.25, 0.5, 0.75, 1\}$. For CQL, we set the parameter $\alpha = 0.1$. Each game dataset contains a fixed number of policies and episodes per policy, resulting in total of 10,000 episodes. This results in varying number of data points per game, and as a result, different total numbers of gradient steps. For clarity, we plot the scaling curves using fraction of training gradient steps on the x-axis, as needed.

4.1. Scaling Plots

The limit of performance in the infinite-data setting for a particular model size gives the asymptotic limit as compute and data are scaled. In each setting, for each algorithm, we run this experiment at several model sizes. Results in several settings are visualized in Figure 4, while more extensive results are given in Appendix A (Figures 8,10).

From these plots, we can see how these algorithms scale with compute and data, as well as with increase scale. In general, our results are quite consistent: almost all algorithms benefit from additional training in almost all settings. One notable exception is in the case of small RCI models trained on low-expertise data, especially when conditioned on returns larger than seen in training, e.g. in Figure 9.

4.2. Algorithm Comparison

We can also compare the performance of various algorithms as we adjust the quality of the data. In Figure 1, we compare the performance of our algorithms, when scaled to convergence in both parameters and compute/data, on each of our experimental settings.

The clearest trend is that the performance-at-scale of these algorithms is closely tied to the performance of the best data-generating expert. Although conditional imitation can sometimes surpass the best expert slightly, neither algorithm tested in the PyBullet domain is consistently capable of surpassing the best expert. In contrast, on the Atari domain, well-tuned CQL consistently outperforms the best expert in the dataset by a significant margin, especially at scale and at low expertise. However, the performance-at-scale of CQL at any fixed expertise level still seemingly reaches an asymptote in all settings, at which point a further increase in performance requires an increase in dataset quality.

In Atari, RCI still reaches a performance-at-scale asymptote at all expertise levels, but uniquely among our results, this asymptote does not consistently increase with improved data quality. We plan to investigate this surprising result further in the future.

4.3. Scaling Model Size vs Dataset Quality

Consider a practical scenario where one has a particular budget to spend on improving the performance of a model. Should that budget be spent on a larger model, or higher-quality data? In this section, we visualize the relative rate of improvement between scaling up the model size, and improving the quality of the dataset by improving the quality of the data-generating expert.

In Figure 2, we can see that in this setting, the performance gained by improving the expertise level of a dataset far exceeds the performance gained by scaling up the model size. In most cases, a relatively modest increase in the performance of the best data-collection expert has greater impact on performance of the trained model than multiple orders of magnitude increase in model size. Results for two settings are given in Figure 2, while more extensive results are given in Appendix A. We note that the scale of effect is

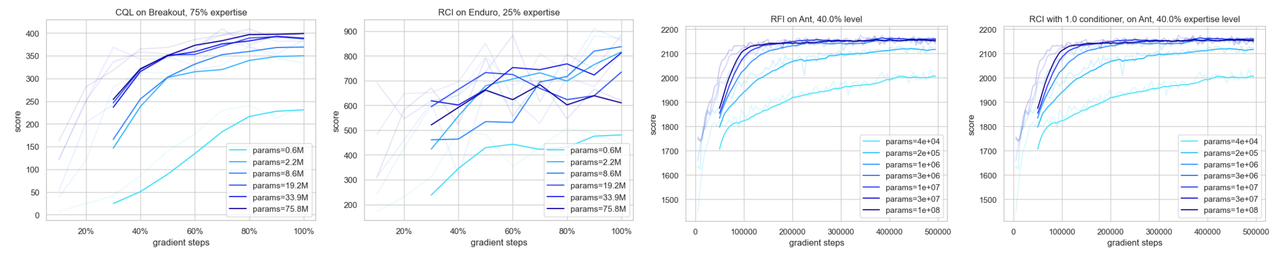

Figure 4. Data, compute, and parameter scaling for algorithms on PyBullet and Atari. All runs use 8 seeds.

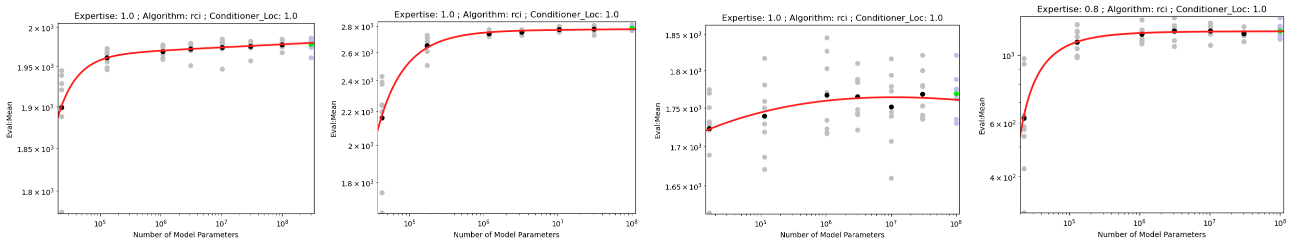

Figure 5. Scaling Laws for RCI HalfCheetah, Ant, Hopper, Walker2D respectively

likely to be algorithm and domain specific, but expect the general principle to be broadly applicable.

4.4. Ablation On Conditioning

An important choice for the RCI algorithm is what return to condition on at inference-time. If the requested return is low, the performance will be low, but if the requested return is too high, generalization may fail, leading to a complete performance collapse. In this section, we study the effect of different conditioning values for the Return Conditioned Imitation Learning algorithm.

In Figure 3, we highlight the results for the RCI algorithm for HalfCheetah and Ant, while more extensive results are given in Appendix A.

It can be seen from these plots that conditioning on returns which are less than the best expert in the dataset behaves as requested, and such, performance gradually increases as the conditioner is increased. However, beyond the best expert, where we rely completely on extrapolative generalization, this trend ends. Performance when conditioned outside of the training range is noisy and unpredictable, oftentimes collapsing, and occasionally increasing. As the conditioner increases, this sensitivity worsens. Overall, these results do not indicate that conditioning higher than the best expert is a reliable way to improve performance, even at the limit of scale.

5. Fitting Scaling Laws

Given the reliable scaling behavior found in supervised learning (Kaplan et al., 2020), it is natural to explore whether it is possible to extrapolate the performance of scaling data, compute, and parameters under a fixed level of data quality. If this is true, it becomes plausible to use smaller-scale runs to estimate the limiting behavior of larger runs; for example, as a low-cost way to determine whether to scale to larger models, or spend resources improving data expertise.

To this end, we explore whether our performance-at-scale results could have been predicted by scaling laws. We fit Broken Neural Scaling Laws (BNSL) (Caballero et al., 2022) given by the functional form:

$$y = a + \left(bx^{-c_0}\right)\prod_{i=1}^{k}\left(1+\left(\frac{x}{d_i}\right)^{1/f_i}\right)^{-c_i*f_i}$$

To fit the above functional form we find the constants $(a, b, c_0, c_1...c_k, d_1...d_k, f_1...f_k)$ that minimize the MSLE, mean squared log error between the predictions (i.e. the y values) of BNSL and the experimental returns. We first perform a grid search using scipy.optimize.brute to find the initial values. These values are then used as initialization for the nonlinear least squares (NLS) algorithm.

We use the numerically stable variant of MSLE for the above optimization given by:

$$MSLE = \sum_{i=1}^{n}((log(y_i+1) - log(\hat{y}_i+1))^2)/n$$

Overall, we find that this procedure is successful, and we are able to extrapolate to the performance of larger models in a variety of settings. Several examples of scaling curves fit by our model are given in Figure 5, and further examples in Appendix A (Figure 12, 11,13,14).

6. Discussion and Conclusions

We have shown via experiments across multiple domains that suboptimal data can bottleneck scaling. This result is, to a certain extent, unsurprising: it seems obvious that without high-quality data on a particular task, an agent is inevitably limited in its ability to perform that task. However, the impressive capabilities of highly-scaled deep neural networks give reason for doubt. Powerful generalization has recently been found to be capable of surprising feats, and it was possible that the same would be true here. However, our experiments confirm the intuition that, indeed, near-expert-level data is required for expert-level performance, even as deep-learning-based agents are scaled to their limits of compute, parameters, and dataset size.

Hoffmann et al. (2022) demonstrated that efficient scaling requires increasing the compute, data size, and parameters at the correct ratio. If one of these factors is increased more slowly, it will bottleneck improvement, leading to diminishing returns from scale on the other factors. Our conclusions can be interpreted as an extension of these results, showing that that data quality is another important dimension of variation that behaves in this the same way. If data quality is not sufficiently high, this will bottleneck learning and impede returns to scale on compute, data size, and parameters.

Currently, large language models are not trained with any particular task in mind. However, it is straightforward to define any number of downstream tasks using natural language. Models tested on tasks demonstrate good scaling with respect to these goals. Our work indicates that these Internet-trained models may at some point become bottlenecked by data quality on some of these tasks, if there is not enough expert data for that task available on the Internet.

When bottlenecked by any of compute, parameters, or data size, there is a clear path forward. But when the bottleneck is expert data, the road is less clear. One way to solve this problem is to simply find a high-performing expert: for example, in offline reinforcement learning, one might hire expert humans to complete the task, record their actions, and add this to the dataset. However, this approach is not scalable. A more autonomous method would leverage the knowledge of model itself to source its own expert data. If our offline RL agent learns a policy which outperforms the best data-collection expert in its dataset, then by interacting according to its policy and adding that data to the dataset, the quality of its dataset will be improved. If this feat can be repeated, the agent can bootstrap itself into arbitrarily-high performance. This, of course, is precisely the setting of online reinforcement learning (Sutton & Barto, 2018). Thus, one interpretation of our contribution is to motivate the online reinforcement learning setting, in light of the impressive capabilities of supervised learning models trained at scale.

References

Agarwal, R., Schuurmans, D., and Norouzi, M. An optimistic perspective on offline reinforcement learning. In *International Conference on Machine Learning*, pp. 104–114. PMLR, 2020.

Agarwal, R., Schwarzer, M., Castro, P. S., Courville, A., and Bellemare, M. G. Beyond tabula rasa: Reincarnating reinforcement learning. *Advances in neural information processing systems*, 2022.

Aghajanyan, A., Yu, L., Conneau, A., Hsu, W.-N., Hambardzumyan, K., Zhang, S., Roller, S., Goyal, N., Levy, O., and Zettlemoyer, L. Scaling laws for generative mixed-modal language models. *arXiv preprint arXiv:2301.03728*, 2023.

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

Bellman, R., Glicksberg, I., and Gross, O. On the "bang-bang" control problem. *Quarterly of Applied Mathematics*, 14(1):11–18, 1956.

Brandfonbrener, D., Bietti, A., Buckman, J., Laroche, R., and Bruna, J. When does return-conditioned supervised learning work for offline reinforcement learning? *arXiv preprint arXiv:2206.01079*, 2022.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.

Buckman, J., Gelada, C., and Bellemare, M. G. The importance of pessimism in fixed-dataset policy optimization. *arXiv preprint arXiv:2009.06799*, 2020.

Caballero, E., Gupta, K., Rish, I., and Krueger, D. Broken neural scaling laws. *arXiv preprint arXiv:2210.14891*, 2022.

Chen, L., Lu, K., Rajeswaran, A., Lee, K., Grover, A., Laskin, M., Abbeel, P., Srinivas, A., and Mordatch, I. Decision transformer: Reinforcement learning via sequence modeling. *arXiv preprint arXiv:2106.01345*, 2021.

Coumans, E. and Bai, Y. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2021.

Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Goodfellow, I., Bengio, Y., and Courville, A. *Deep learning*. MIT press, 2016.

Gulcehre, C., Wang, Z., Novikov, A., Paine, T. L., Colmenarejo, S. G., Zolna, K., Agarwal, R., Merel, J., Mankowitz, D., Paduraru, C., et al. Rl unplugged: Benchmarks for offline reinforcement learning. *arXiv preprint arXiv:2006.13888*, 2020.

Hessel, M., Modayil, J., Van Hasselt, H., Schaul, T., Ostrovski, G., Dabney, W., Horgan, D., Piot, B., Azar, M., and Silver, D. Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., Casas, D. d. L., Hendricks, L. A., Welbl, J., Clark, A., et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Hussein, A., Gaber, M. M., Elyan, E., and Jayne, C. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50(2):1–35, 2017.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative q-learning for offline reinforcement learning. *arXiv preprint arXiv:2006.04779*, 2020.

Kumar, A., Agarwal, R., Geng, X., Tucker, G., and Levine, S. Offline q-learning on diverse multi-task data both scales and generalizes. *International Conference on Learning Representations*, 2023.

Laroche, R., Combes, R. T. d., and Buckman, J. Non-markovian policies occupancy measures. *arXiv preprint arXiv:2205.13950*, 2022.

LeCun, Y., Bengio, Y., et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

Levine, S., Kumar, A., Tucker, G., and Fu, J. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Maloney, A., Roberts, D. A., and Sully, J. A solvable model of neural scaling laws. *arXiv preprint arXiv:2210.16859*, 2022.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *nature*, 518(7540): 529–533, 2015.

Nakkiran, P., Kaplun, G., Bansal, Y., Yang, T., Barak, B., and Sutskever, I. Deep double descent: Where bigger models and more data hurt. *Journal of Statistical Mechanics: Theory and Experiment*, 2021(12):124003, 2021.

Raffin, A., Hill, A., Ernestus, M., Gleave, A., Kanervisto, A., and Dormann, N. Stable baselines3, 2019.

Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., and Chen, M. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

Robbins, H. and Monro, S. A stochastic approximation method. *The annals of mathematical statistics*, pp. 400–407, 1951.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

Schmidt, D. and Schmied, T. Fast and data-efficient training of rainbow: an experimental study on atari. *arXiv preprint arXiv:2111.10247*, 2021.

Schuhmann, C., Beaumont, R., Vencu, R., Gordon, C., Wightman, R., Cherti, M., Coombes, T., Katta, A., Mullis, C., Wortsman, M., et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *arXiv preprint arXiv:2210.08402*, 2022.

Seyde, T., Gilitschenski, I., Schwarting, W., Stellato, B., Riedmiller, M., Wulfmeier, M., and Rus, D. Is bang-bang control all you need? solving continuous control with bernoulli policies. *Advances in Neural Information Processing Systems*, 34:27209–27221, 2021.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Chi, E., Le, Q., and Zhou, D. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022.

Štrupl, M., Faccio, F., Ashley, D. R., Schmidhuber, J., and Srivastava, R. K. Upside-down reinforcement learning can diverge in stochastic environments with episodic resets, 2022.

A. Additional Figures

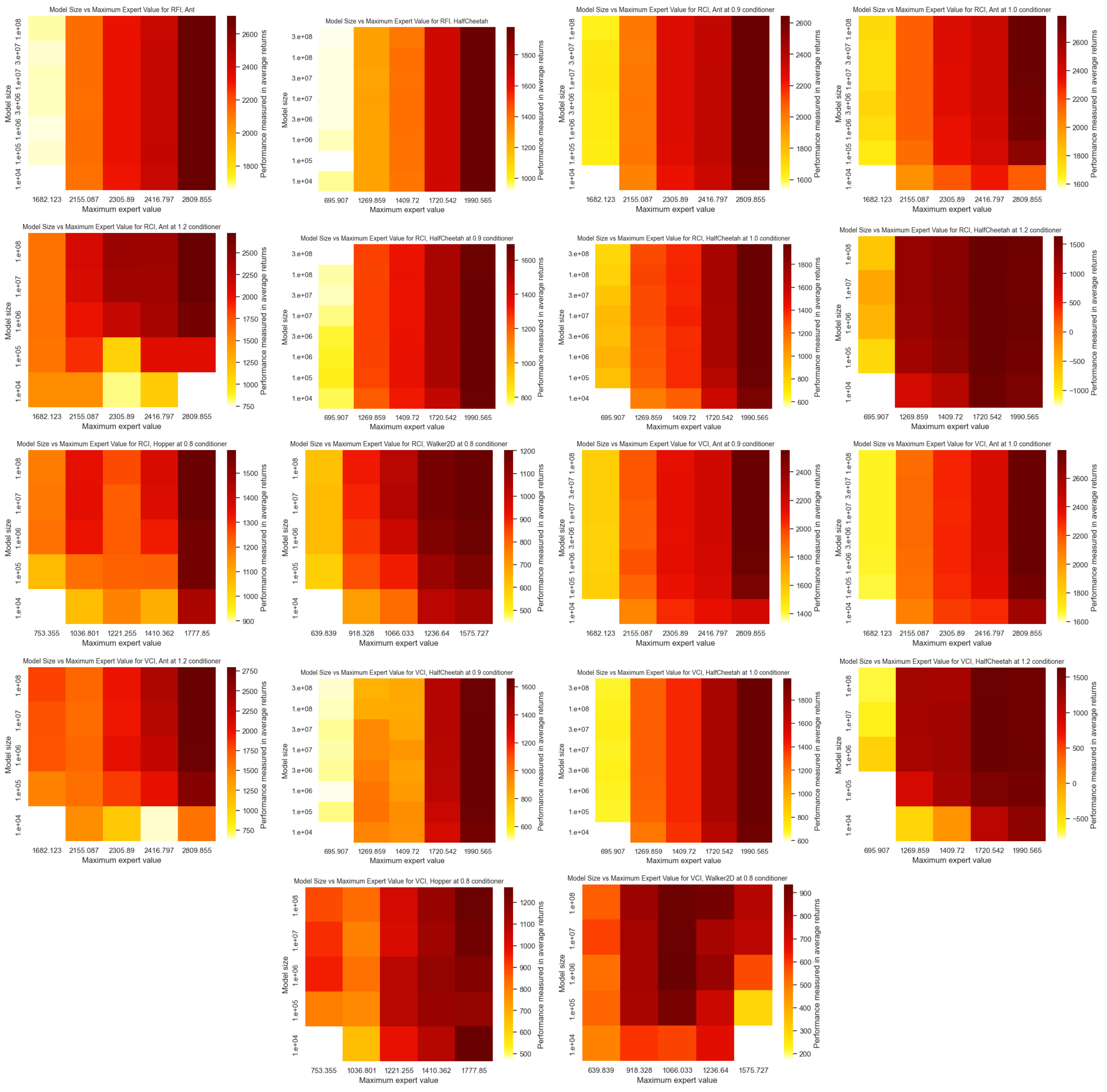

Figure 6. Tradeoff between model size and dataset quality

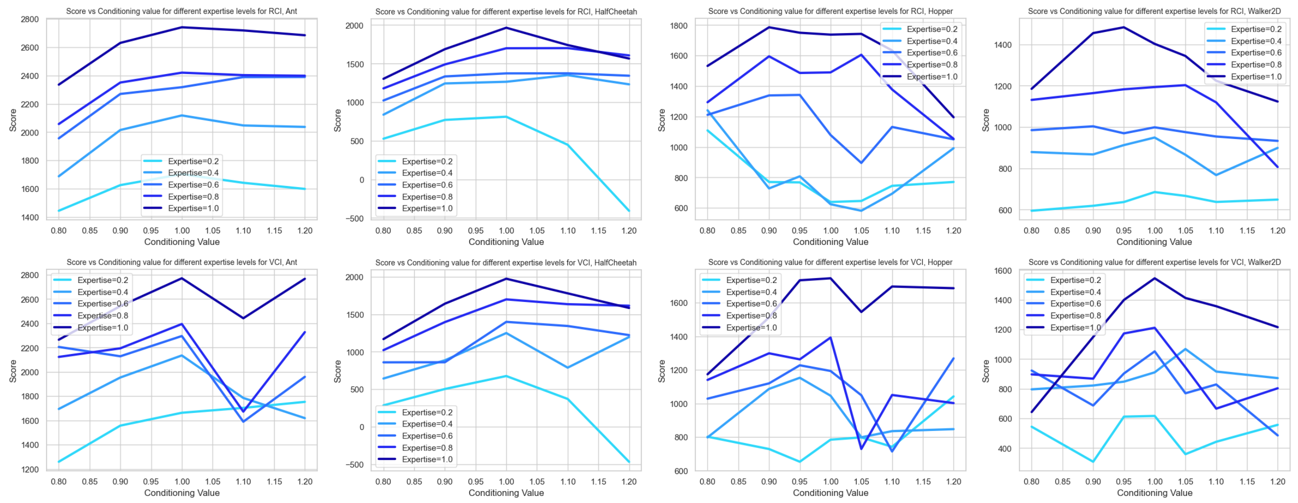

Figure 7. Ablation study for different conditioning values for RCI and VCI

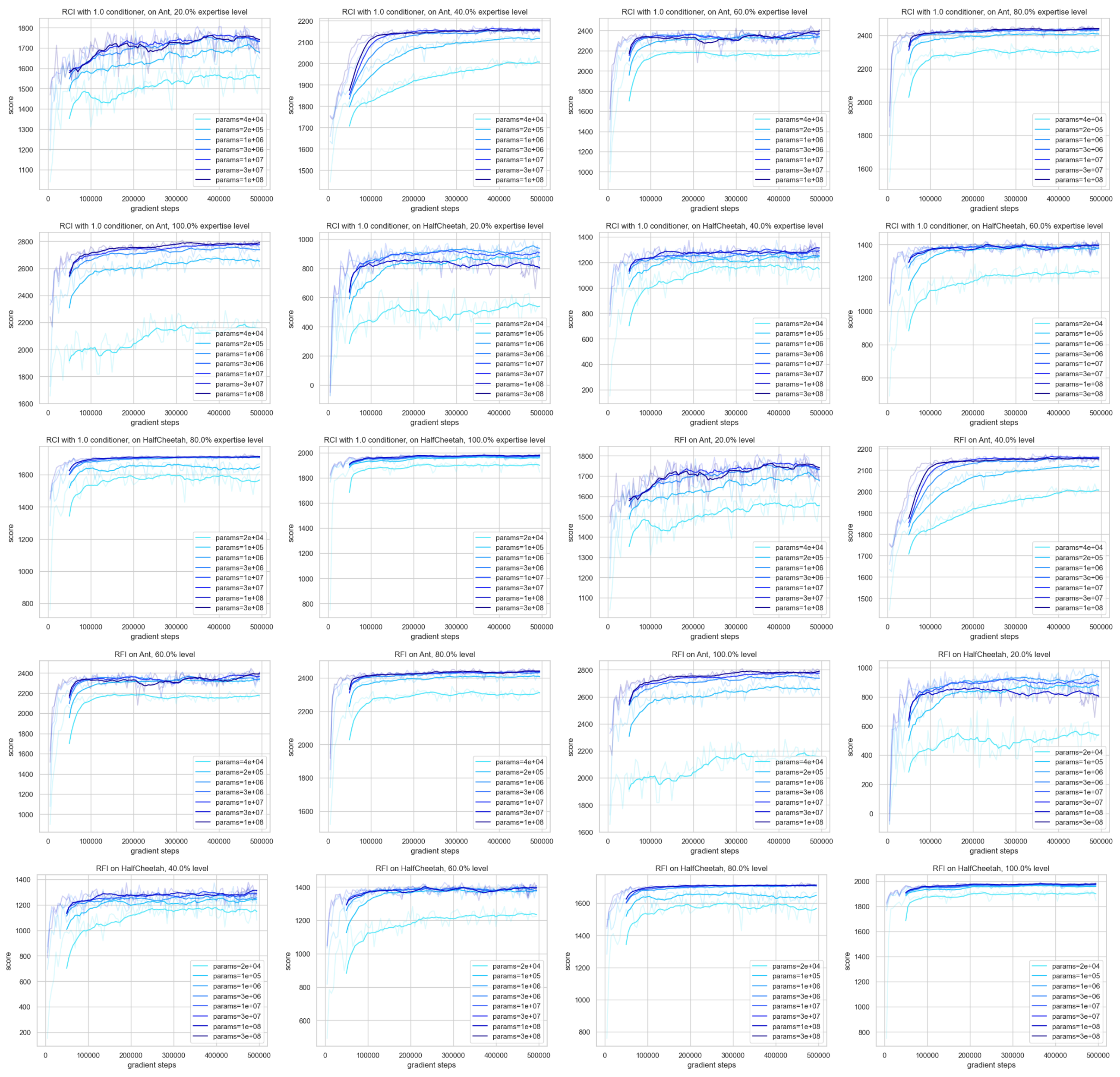

Figure 8. Scaling curves for PyBullet for RCI and RFI

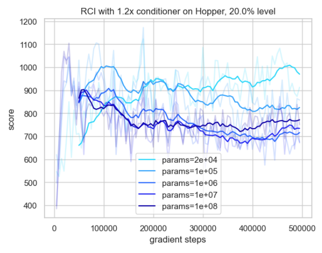

Figure 9. Inverse scaling in Hopper

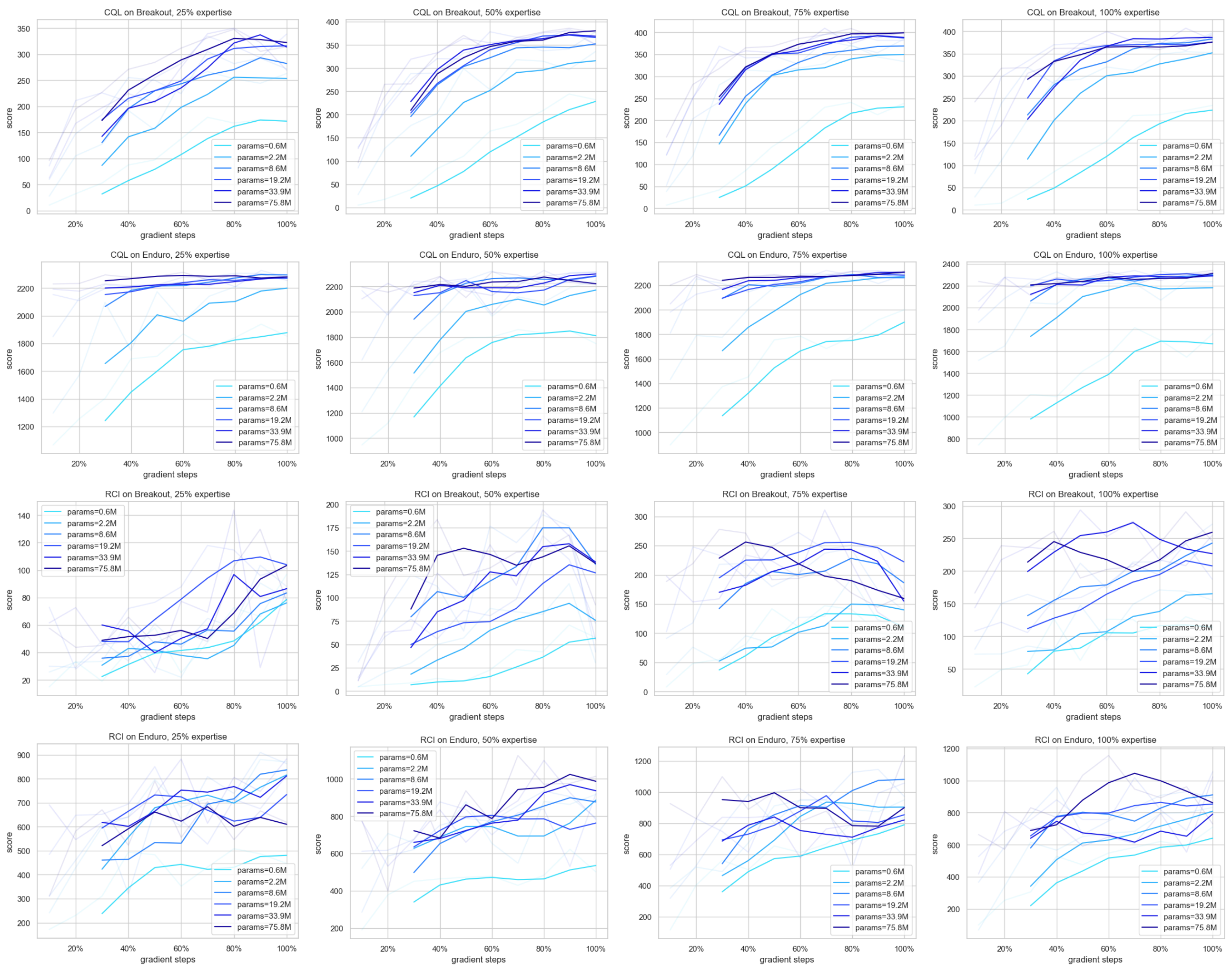

Figure 10. Scaling curves for Atari for RCI and CQL

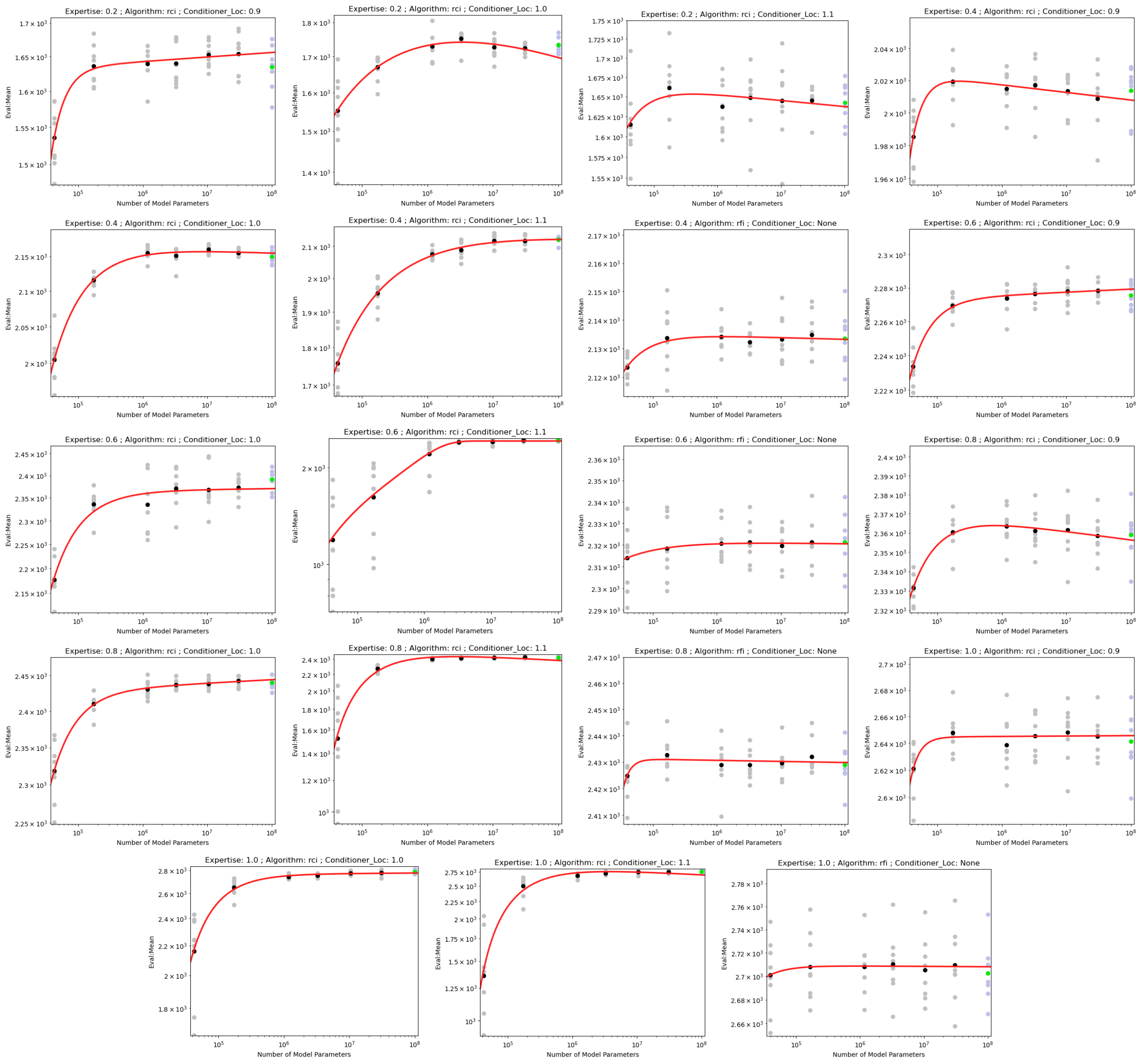

Figure 11. Broken Neural Scaling Laws fits and extrapolations for Ant. Black points are mean of gray points at each model size. Red line is BNSL fit to black (or technically gray too) points. Light blue points are held out to evaluate extrapolation, and green points are the mean of the light blue points.

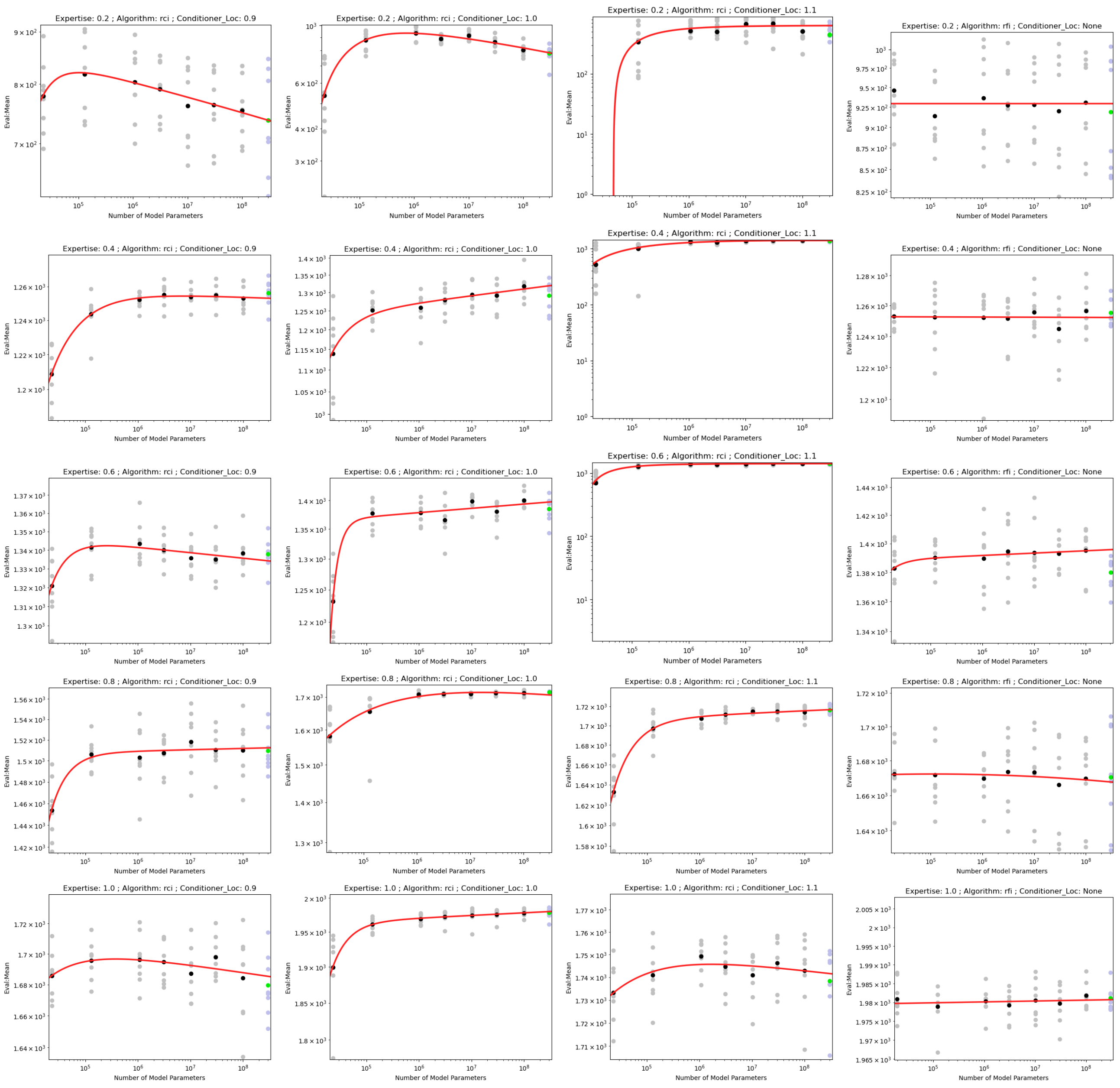

Figure 12. Broken Neural Scaling Laws fits and extrapolations for Half Cheetah. Black points are mean of gray points at each model size. Red line is BNSL fit to black (or technically gray too) points. Light blue points are held out to evaluate extrapolation, and green points are the mean of the light blue points.

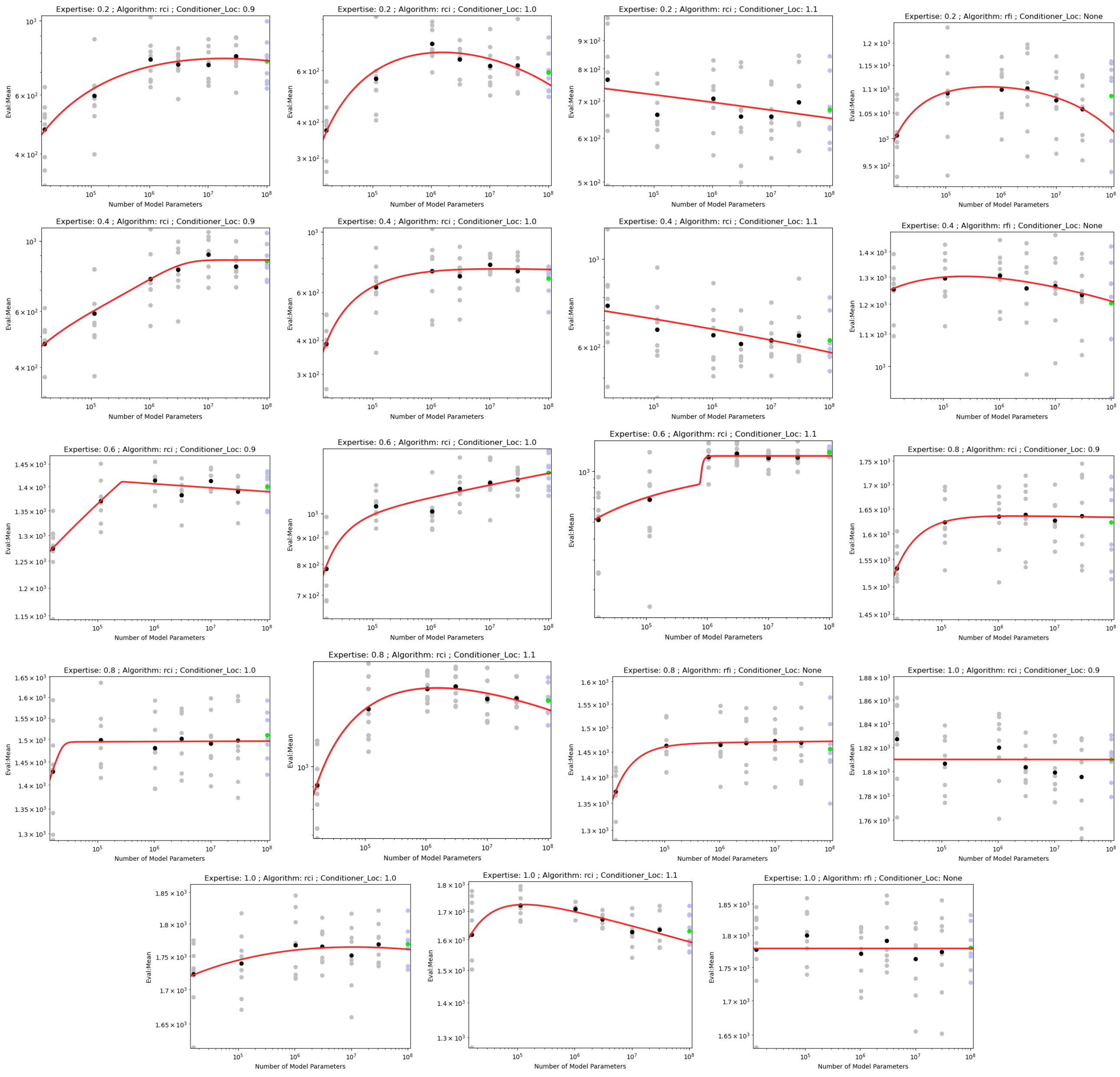

Figure 13. Broken Neural Scaling Laws fits and extrapolations for Hopper. Black points are mean of gray points at each model size. Red line is BNSL fit to black (or technically gray too) points. Light blue points are held out to evaluate extrapolation, and green points are the mean of the light blue points.

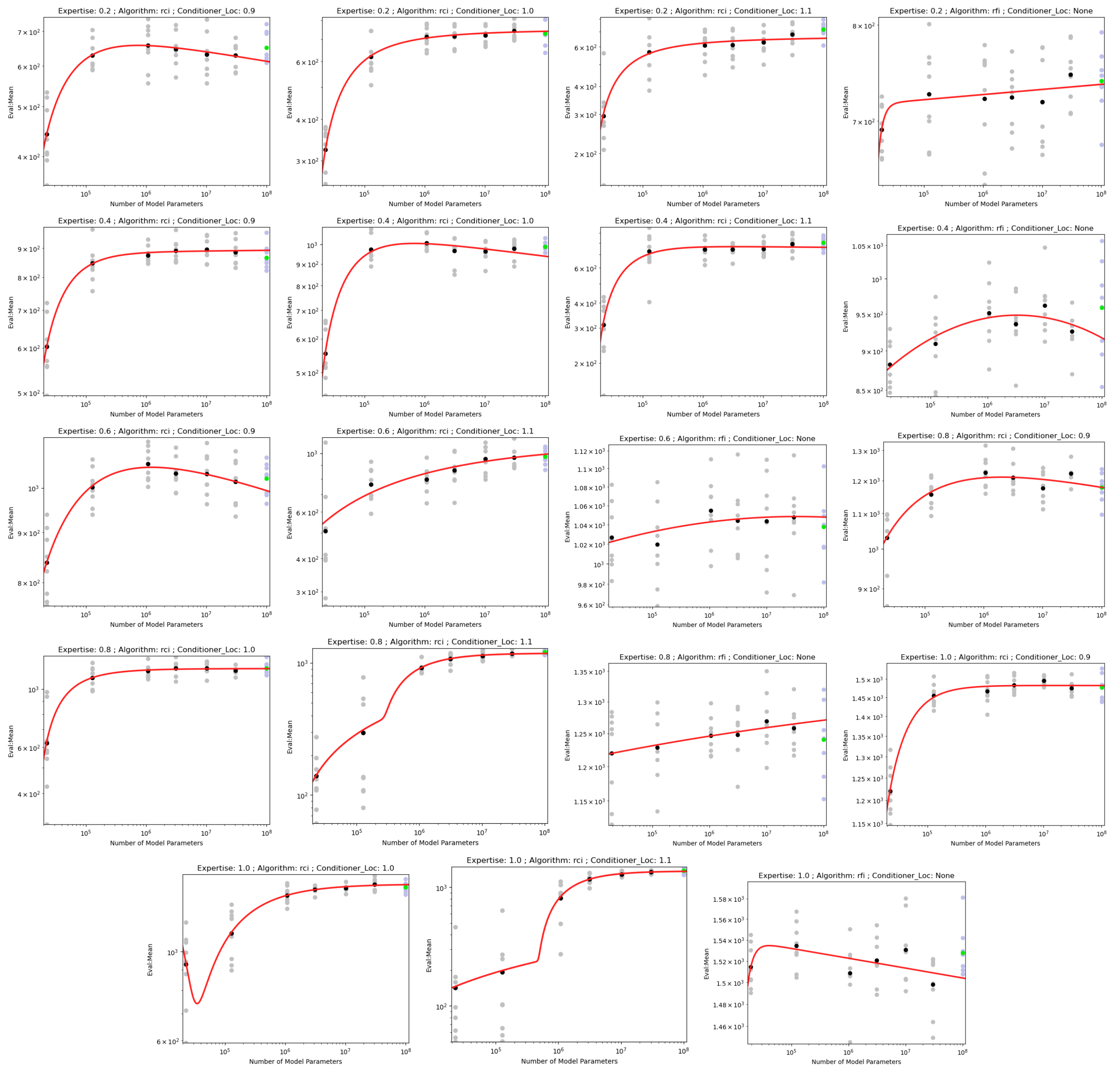

Figure 14. Broken Neural Scaling Laws fits and extrapolations for Walker2D. Black points are mean of gray points at each model size. Red line is BNSL fit to black (or technically gray too) points. Light blue points are held out to evaluate extrapolation, and green points are the mean of the light blue points.