Extended Abstract

Motivation Reinforcement learning is still too sample inefficient to be practically useful for real-world generalist robot tasks with sparse rewards, and human expert demonstrations for behavior cloning are very expensive. Using a small number of expert demonstrations to substantially improve online RL's sample efficiency is thus desirable. However, existing approaches either regularize the actor toward the demo distribution (potentially constraining it too strongly) or simply oversample demos from the replay buffer, leaving the crucial exploration phase uninformed by expert data. More recent approaches use BC reference policies to guide exploration and bootstrap updates, but these may not be robust to data quality variation or issues tuning BC policy training.

Method Data-Guided Noise (DGN) is a novel approach that attempts to implicitly guide online exploration by identifying the axes of the action space on which the RL policy tends to make errors relative to expert actions. By exploring more widely along these axes, DGN targets its exploration, improving sample efficiency of RL algorithms using rollout data collected with DGN. More concretely, DGN learns the policy mean with RL and models the policy covariance around that mean with imitation learning. This covariance around the policy mean captures the axes on which the RL policy actions differ most from expert actions. Sampling actions from the multivariate Gaussian centered on the policy mean with this learned covariance thus entails exploring more widely along these axes (on which the RL policy is making errors relative to the expert). The learned covariance is state-conditioned, parameterized by an MLP that maps from observations to entries of the (Cholesky decomposition of) the covariance matrix.

Implementation In this work DGN is implemented on top of an RLPD base algorithm (Ball et al., 2023). DGN is compared to several baseline online RL algorithms that utilize expert demonstrations (including algorithms that use BC regulariation terms, that oversample demos from the replay buffer, and that use a BC reference policy). These comparisons are made on Robosuite (Mandlekar et al., 2021) tasks known to be difficult for RL due to their long horizons, requirements of precision, and sparse 0/1 rewards.

Results DGN demonstrates higher sample efficiency than baselines that oversample demos from the replay buffer or use BC regulization losses on three out of four tasks. DGN slightly underperforms a method using a BC reference policy under optimal conditions but is shown to be more robust to lower expert data quality or high dataset multimodality.

Discussion DGN's performance relative to other methods is larger on more difficult tasks. This is a promising sign that DGN will continue to be a useful method as the field scales to harder tasks and in a scaled-up industry setting. A key limitation of DGN is that, while on the whole it improves sample efficiency, it does not seem to help sample efficiency towards the end of an RL training run, once the RL policy is already highly performant. However, this may be a fundamental limitation of any method that uses a limited expert dataset; at some point, the increasing amount of information gained online will be better than the fixed expert dataset.

Conclusion DGN shows better sample efficiency relative to many commonly used online RL algorithms for robot policy learning. That higher sample efficiency is shown to be robust to real world issues such as data quality and multimodality. Combined, this suggests DGN may help make hybrid methods improving online RL with expert demos competitive with the purely imitation-based approaches common in the most performant systems demonstrated today.

Data-Guided Noise (DGN) for Online Exploration

Alec Lessing
Department of Computer Science
Stanford University
aleclessing@cs.stanford.edu

Abstract

Reinforcement learning is still too sample inefficient to be practically useful for real-world generalist robot tasks with sparse rewards, making using a small number of expert demonstrations to substantially improve online RL's sample efficiency an attractive alternative approach. Data-Guided Noise (DGN) is a novel approach that implicitly guides online exploration by learning the policy mean with RL and modeling a state-conditioned covariance around that mean with imitation learning. Then, by by sampling exploration actions from the resulting multivariate Gaussian, the agent explores more widely along the axes on which the RL policy tends to make errors relative to expert actions. Implemented on top of the RLPD base algorithm, DGN matches or outperforms baselines that regularize the actor toward the demo distribution, oversample demos from the replay buffer, or use a BC reference policy on Robosuite tasks known to be difficult due to long horizons, precision requirements, and sparse 0/1 rewards. DGN is shown to be more robust to lower expert data quality or high dataset multimodality. Its performance gains are larger on more difficult tasks, though the benefit tapers toward the end of training once the RL policy is already highly performant, likely because the fixed expert dataset is eventually outweighed by online experience. Overall, DGN's robust sample-efficiency improvements suggest that hybrid methods augmenting online RL with expert demos can compete with purely imitation-based approaches in practical robotic settings.

1 Introduction

Currently, in real-world end-to-end robot policy learning, pure behavior cloning requires a large number of expert demonstrations to solve, at a reasonable level of reliability, even on narrow tasks that are complicated and long-horizon enough to require a generalist robot. While data collected through RL can be, in a sense, “cheaper" since it does not require an expert human demonstrator and can leverage suboptimal data and not just high-quality expert demos, much more data is generally required to use RL effectively. Hybrid methods, using a smaller number of expert demos to allow for more sample efficient RL, may thus be a desirable alternative.

The core objective of this project is to maximize the *sample efficiency* of off-policy, online RL for robot policy learning when the agent *has access to a small number of expert demonstrations* by using these expert demonstrations to make exploration more targeted.

Numerous prior works attempt to use expert demonstrations to improve sample efficiency in online RL. One line of work simply adds an imitation learning regularization objective to an RL algorithm (Nair et al., 2018). While these approaches can help performance early on, they can too strongly constrain the RL policy towards the expert distribution, limiting improvements in performance. In addition, there can be mismatches between RL objectives and imitation learning objectives, leading to poor optimization of both objectives. Other approaches such as RLPD include expert demonstrations in the replay buffer and oversample them during training (Ball et al., 2023). While these methods have been shown to help, they neglect to use the expert demonstration data during the exploration portion of RL, leaving potential performance gains on the table. Recently, methods such as IBRL (Hu et al., 2024) use reference policies trained with imitation learning to guide RL exploration. However, as shown in section 5.2, these methods may not be robust to varying expert data quality and multimodality of the expert dataset. They also may require substantial resources and evaluations be put into tuning good BC reference policies.

This project introduces Data-Guided Noise (DGN), a method to overcome these limitations of prior methods by implicitly guide RL exploration using expert demonstrations to choose which directions to explore. DGN models the axes on which the RL policy tends to make errors relative to expert actions and upweights exploration along those axes. More mathematically, DGN learns the policy mean with RL and models the policy variance around that mean with imitation of expert demos. This produces exploratory actions that are more informative and thus leads to more sample efficient exploration.

In this report, I will address the following research questions:

1. How does DGN's sample efficiency compare to prior RL methods that utilize expert demos? More precisely, can modifying a standard RL algorithm (RLPD, Ball et al. (2023)) by changing *exploration actions alone* lead to higher sample efficiency?
2. How does DGN's sample efficiency relative to other methods change under different conditions (task difficulty, number of availible demos, ect.)? When is it best to use DGN?
3. How robust is DGN to changes in expert data quality and other data attributes like multimodality? How does its robustness compare to that of state-of-the-art methods?
4. What aspects of DGN (including components and hyperparameters of the methodology) are most important for performance?

2 Related Work

To address limitations of using RL or imitation learning alone, several works have investigated ways to improve online robot RL using expert demonstrations.

A number of works add imitation learning objectives to RL actor updates to regularize the policy towards the expert dataset. Nair et al. (2018) added a behavior cloning loss term to act as a regularizer during policy updates to DDPG and HER (Lillicrap et al., 2019; Andrychowicz et al., 2018). Their experiments showed that this method could induce an order of magnitude speedup over pure RL on simulated robot tasks. However, while this sort of approach often helps performance early on in training, it often constrains the policy too closely to the expert dataset, limiting performance later (Nair et al., 2020). In contrast, DGN only indirectly encourages imitation of the expert demos, upweighting exploration away from the policy mean along various axes using the expert demonstrations. Since DGN does not explicitly constrain the policy itself, it avoids the issues common to methods that explicitly regularize with a BC loss.

Another line of work adds expert demonstrations to replay buffers and oversamples those demonstrations (Ball et al., 2023; Vecerik et al., 2018). Alongside other subtle tricks, RLPD oversamples expert demonstrations by maintaining separate replay buffers of expert demonstrations and online rollouts and then sampling from them in a one-to-one ratio. Ball et al. (2023) found this trick (alongside a few others) could substantially improve the performance of SAC (Haarnoja et al., 2018) in cases where offline data was available. While these tricks improve sample efficiency using offline data, they neglect to use expert demonstrations at all during the exploration portion of the RL process. As shown in section 5, the sample efficiency of one of these methods, RLPD, can be improved by using expert demos to guide exploration with DGN.

One of the most recent state-of-the-art works using expert demos to improve RL sample efficiency is Imitation Bootstrapped Reinforcement Learning (IBRL, Hu et al. (2024)). IBRL assumes access to a set of expert demonstrations and a BC reference policy trained on those demonstrations. IBRL then modifies TD3 (Fujimoto et al., 2018) by sampling actions from both the fixed BC policy and the RL policy and choosing which action to use as targets for critic updates and as actions to take during rollouts based on which action has a higher associated Q-value. While IBRL does use expert demonstrations to inform exploration, the only direct way which it does so is by sometimes choosing to use to use actions generated by the fixed BC policy during rollouts for data collection. Otherwise, exploration actions are sampled by adding Gaussian noise drawn from a fixed distribution to actions sampled from the deterministic RL policy. This can cause IBRL to be sensitive to the quality of the BC policy. IBRL's BC reference policy must model the expert well in order for the BC policy to provide useful guidance. IBRL can only suggest actions from its BC policy; if the BC reference policy suggests poor actions, IBRL will not perform well. In contrast, DGN models an approximate distribution that it uses to weight the sampling of perturbations to actions. When it cannot model the expert well due to poor dataset quality or imperfect tuning of the learning algorithm, its distribution can simply broaden, still likely encompassing helpful actions perturbation to take. DGN is thus more robust to variation expert dataset quality than IBRL and does not suffer from the need to tune the BC policy.

3 Method

They key idea behind DGN is to use the differences between the RL policy actions and expert demo actions to guide exploration. DGN trains an RL policy π_θ – where θ changes throughout the learning process but notationally represents the current parameters of the RL policy – and assumes access to a dataset of expert demos $\mathcal{D}_{expert}$ containing (s_t, a_t^*) tuples of observed states and expert actions taken in those states.

3.1 Motivating DGN

Mathematically, we can write down the actions the agent takes in the environment during exploration as

$$a_t = \pi_\theta(s_t) + \epsilon_t \tag{1}$$

where s_t is the current observed state and ϵ_t is some noise added to the RL policy action in order to explore. In many RL algorithms ϵ_t will be sampled from $\mathcal{N}(0, \sigma I)$ where σ is a fixed or scheduled hyperparameter or learned from RL actor updates. DGN modifies this by instead learning the distribution from which ϵ_t is sampled using a small expert demo dataset.

Consider the example of a robot with a parallel jaw-gripper picking up a brick. Suppose that the actions are end-effector and gripper positions and suppose that the main failure mode of the current RL policy π_θ is that the robot reaches too high or too low; however, π_θ tends to line up the gripper with the brick precisely in the left/right axis. In this case, it would be sensible to explore more in the up/down direction rather than in the left/right direction. Mathematically, in the context of equation 1, this means sampling noise ϵ_t from a noise distribution that is broad in the up/down direction but narrow in the left/right direction.

DGN aims to learn such a distribution using the expert dataset as a guide. In this example case, because the RL policy π_θ tends to err in the up/down direction, the actions from the RL policy will will differ from expert actions (taken from the same states) more in the the up/down direction than the left/right direction. In other words, the difference $a_t^* - \pi_\theta(s_t)$, where $(s_t, a_t^*) \in \mathcal{D}_{expert}$, will tend to have a large magnitude in the up/down direction and a small magnitude in the left/right direction, precisely because the RL policy π_θ is making more errors in that up/down direction.

The differences $a_t^* - \pi_\theta(s_t)$ are thus a helpful guide for the direction that the agent should explore. Learning a model of the distribution of $a_t^* - \pi_\theta(s_t)$ (for all expert examples) thus will yield the sort of distribution we want to draw ϵ_t from, one that makes ϵ_t biased along the axis that π_θ is currently making errors.

3.2 The Learned DGN Distribution

The learned distribution motivated above in section 3.1 is parameterized as $\mathcal{N}(0, \Sigma_\phi(s))$, where $\Sigma_\phi(s)$ is a learned, state-dependent covariance matrix modeled by a neural network Σ_ϕ with parameters ϕ that maps from observed states s to covariance matrices. A Gaussian distribution with an arbitrary covariance matrix is a suitable choice to parameterize the DGN distribution because it is the simplest model that can model a different scale for exploration noise in every direction. In particular, unlike a model that used a set of independent Gaussians for each dimension of the action space(each with their own independently learned standard deviation), this model can capture arbitrary directions in the action space, allowing exploration to be even more targeted.

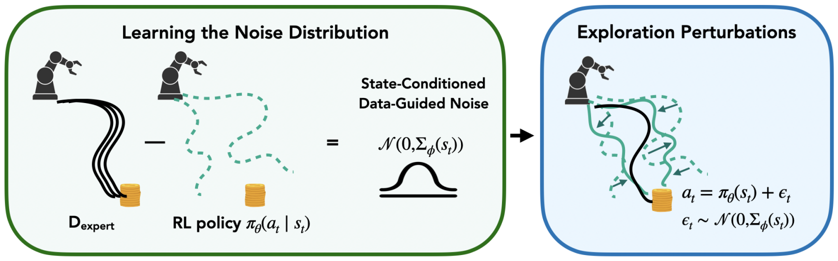
Learning the Noise Distribution
State-Conditioned Data-Guided Noise
$\mathcal{N}(0,\Sigma_\phi(s_t))$
D_expert
RL policy $\pi_\theta(a_t \mid s_t)$
Exploration Perturbations
$a_t = \pi_\theta(s_t) + \epsilon_t$
$\epsilon_t \sim \mathcal{N}(0,\Sigma_\phi(s_t))$

Figure 1: Method Overview. **Left**: DGN learns to model the distribution of differences between expert actions and the actions the RL policy would take in the same states. This distribution is parameterized as a state-conditioned covariance matrix of a zero-centered Gaussian distribution. **Right**: During exploration rollouts, DGN adds noise sampled from the learned model to RL policy actions to decide which actions to take in the environment.

Arbitrary covariance matrices cannot simply be parameterized by arbitrary matrices because covariance matrices must have certain mathematical properties (such as being positive semi-definite and symmetric). Using a trick found commonly in the literature, $\Sigma_\phi(s)$ is not the direct output a neural network. Rather, a neural network maps states to the nonzero entries of a lower triangular matrix $A_\phi(s)$. The `softplus` activation is applied to the diagonal of this matrix to ensure that it is positive. The covariance matrix is then computed as $\Sigma_\phi(s) = A_\phi(s)A_\phi(s)^T$. (Note that, while the covariance matrix $\Sigma_\phi(s)$ can be computed this way, the implementation in code actually uses $A_\phi(s)$ to perform calculations involving $\mathcal{N}(0, \Sigma_\phi(s))$.)

It is important that the distribution parameterized by $\Sigma_\phi(s)$ be state-conditioned. This is because the optimal direction of exploration will very likely be different in different states. To return to the brick-grasping example, perhaps the RL policy tends to err in the up/down direction as the robot grasps the brick but tends to err in the left/right direction when it places the brick. The learned DGN distribution $\mathcal{N}(0, \Sigma_\phi(s))$ must be state-conditioned to capture this. The importance of this design choice is empirically examined in section 5.3 below.

3.3 The DGN Algorithm

DGN modifies RLPD, solely by modifying the way actions to take during exploration rollouts are sampled. The modifications to the RLPD algorithm can be split into two phases (visualized in Figure 1):

- **Learning the DGN Distribution:** Every $N = 1000$ steps of online interaction, the parameters ϕ of the learned distribution $\mathcal{N}(0, \Sigma_\phi(\cdot))$ are trained to minimize the negative log-likelihood loss of the expert demo dataset. Specifically, ϕ is updated to mimize the loss

$$\mathcal{L}_{DGN} = \sum_{(s_t, a_t^*) \in \mathcal{D}_{expert}} \left[-\log p_\phi \left(a_t^* - \text{s.g}(\pi_\theta(s_t)) \middle| s_t \right) \right] \tag{2}$$

  where $p_\phi(\cdot|s_t)$ is the density of distribution $\mathcal{N}(0, \Sigma_\phi(s_t))$ and s.g. denotes the stop-gradient operator (meant to emphasize that only ϕ and not ϕ is updated with this loss).

  These updates to the learned DGN distribution are done continuously throughout the RL learning processs (every $N = 1000$ steps) because the current RL policy π_θ changes over the course of training. The kinds of mistake that π_θ makes thus also change over the course of training and the learned DGN distribution must be continuously updated to model this.

- **Exploring with the Learned DGN Distribution:** At each environment step during exploration, actions are chosen by perturbing RL policy actions using noise sampled from the DGN learned noise distribution.

$$a_t = \pi_\theta(s_t) + \epsilon_t \quad \text{where} \quad \epsilon_t \sim \mathcal{N}(0, \Sigma_\phi(s_t)) \tag{3}$$

Note that, mathematically, the *only difference* between DGN and RLPD is that, in RLPD, the noise ϵ_t is sampled from $\mathcal{N}(0, \sigma I)$ where σ is a fixed or scheduled hyperparameter or learned from RL actor updates.

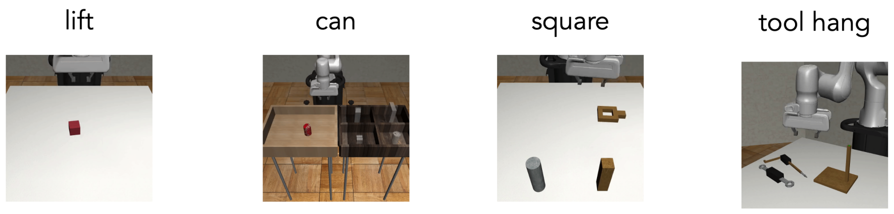

Figure 2: Task Environments: The four robosuite (Mandlekar et al., 2021) tasks used to evaluate DGN and baselines in this project. Roughly speaking, task difficulty and horizon increases from left to right.

3.4 Shutting Off DGN

Once the RL policy being trained by DGN reaches a high level of performance (specifically a 50% success rate, measured with the previous ten exploration episodes), DGN is shut off. DGN can substantially increase sample efficiency early on, when the RL policy is poor and can benefit from implicit guidance from expert demonstrations. However, once the RL policy already does quite well, it may already have more information than is captured in the expert dataset. At this point, using expert demonstrations to target exploration may no longer be beneficial. The 50% success rate threshold is a simple way of roughly identifying when this is the case.

4 Experimental Setup

4.1 Tasks

Environments: Four tasks (shown in Figure 2) from the robosuite suite of tasks were used to evaluate DGN and baselines for this project:

- **Lift:** The objective is to pick up the square. 100 demonstrations were used for this task.
- **Can:** The objective is to pick up the can from one table and place it in one of the zones on the other table. 50 demonstrations were used for this task.
- **Square:** The objective is to pick up the square nut and put it on the square peg. 50 demonstrations were used for this task.
- **Tool Hang:** The objective is to assemble a stand by inserting it vertically into the base and then hang a tool from the stand. 200 demonstrations were used for this task.

Mandlekar et al. (2021) provides both these simulated environments and human demonstrations of each of these tasks. Each of these environments has a sparse 0/1 reward signaling task failure of completion at the end of each episode. This reward function, alongside the time horizon of these tasks and precision required in several of them, is known to make these tasks difficult to solve with reinforcement learning (Mandlekar et al., 2021; Hu et al., 2024).

Observation and Action Spaces: The observation space of each task consisted of end-effector pose, gripper position, and the pose (and relative pose) of objects in the environment relative to the tasks. Observations included three steps of history from the most recent environment steps. The action space consisted of relative end-effector pose and gripper position.

4.2 Algorithm and Architecture:

Policy Architectures: The policies and DGN models used the same architecture for each task. Policies were MLPs with three hidden layers, each with 1024 hidden units. The learned DGN distribution (the neural network that maps states to covariance matrices) was parameterized as an MLP with two hidden layers of 128 hidden units.

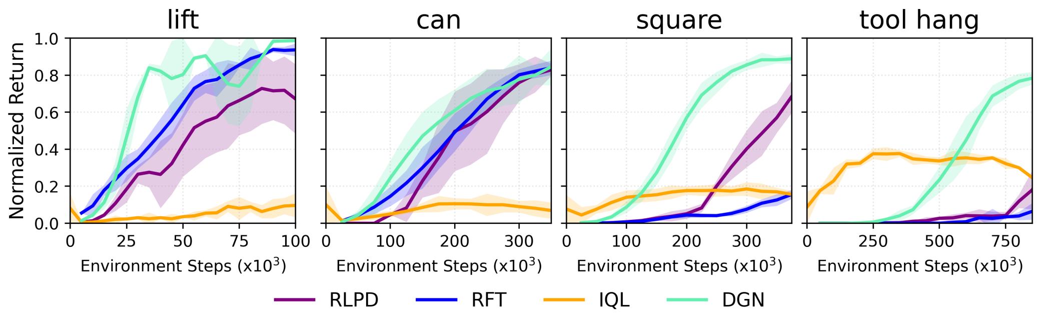

Figure 3: Learning Curves (normalized returns by number of environment timesteps) for DGN and several baselines. DGN matches or outperforms each baseline on all tested tasks. The relative performance of DGN was best on the hardest, longest-horizon tasks: `square` and `tool hang`.

4.3 Baselines:

In experiments, DGN is compared to several baseline methods that (can) use expert demonstrations to improve sample efficiency in online RL:

- **IBRL** (Imitation Bootstrapped RL) first trains a BC reference policy on expert demos. During online RL, chooses between an action sampled from the BC reference policy and the RL policy using the relative Q-values of each action. These actions are used both for exploration and for Bellman backups. (More details above in Section 2).
- **RLPD** is a modification of SAC which oversamples expert demonstrations from the replay buffers during online training, effectively upweighting the expert data (Ball et al., 2023). The version of SAC used as a base algorithm in (Ball et al., 2023) uses hyperparameters that make it effectively close to TD3 (Fujimoto et al., 2018), which is used the the base algorithm for RLPD in this work.
- **RFT** (Regularized Fine-Tuning) warm-starts the actor by pretraining on expert demos with a BC loss, then adding a BC loss term to actor loss during online RL (with a TD3 base algorithm).
- **IQL** finetuning uses expectile regression to learn a value function that models the 80% expectile and then extracts a policy using AWR (Kostrikov et al., 2021; Peng et al., 2019).

In all baselines, noise drawn from a fixed distribution $\mathcal{N}(0, 0.1)$ was added to actions taken in the environment for exploration. The same was done for DGN after the shutoff described in section 3.4.

4.4 Further Experimental Details

Evaluation: The main metric used for evaluation in this work is the success rate of a learned policy on a task. Specifically, the number of environment steps of exploration needed for the policy to reach a given level of performance is of interested. The relative success rates of policies trained with different methods for the same number of steps is similarly of interest. The learning curves presented below allow for comparison with these metrics.

Each learning curve in section 5 shows the mean and standard error across at least three runs seeds. One hundred evaluation episodes were used to produce these learning curves. These evaluation episodes did not contribute data to the RL process and merely benchmarked it intermittently throughout the learning process. During these evaluation episodes, actions equal to the policy mean (without adding any noise from DGN or any other method) were executed in the environment.

Implementation: These experiments were implemented in a fork of the code published by Hu et al. (2024).

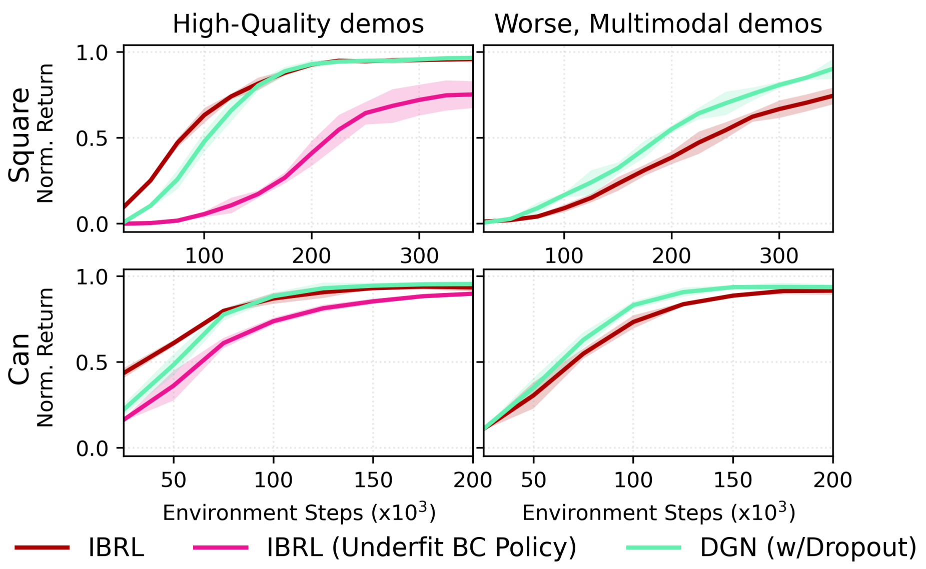

Figure 4: Left: Comparison of DGN to IBRL and IBRL with a poorly fit reference policy on a high-quality dataset without much multimodality. Right: Comparison of DGN to IBRL on a lower-quality, multimodal dataset. While IBRL outperforms DGN under optimal conditions, DGN is more robust to dataset quality and does not suffer from having to tune a BC policy well.

5 Results and Analysis

5.1 Comparison to Standard Baselines

Figure 3 shows learning curves for DGN and several standard baselines in each of the four evaluated tasks. It is immediately evident from these learning curves that DGN is more or equally sample efficient when compared to each of the baselines. The relative performance of DGN is best on the hardest tasks – `square` and `tool hang` – which are longer horizon and require the most precision to complete.

These results, particularly the comparison of DGN to RLPD, validate the core hypothesis that using DGN to target exploration improves sample efficiency. The only difference between the algorithm used in the "DGN" and "RLPD" experiments in Figure 3 is the use of DGN to change exploration actions. Everything else, such the algorithm used to learn from the data in the rollout buffers, is the same. This shows that DGN does improve exploration: fewer environment steps worth of rollout data are required to achieve a given level of performance.

Because DGN explores along the axes in which the RL policy differs from the expert, the data it collects during exploration allows the RL algorithm to more quickly correct the “errors" the RL policy makes relative to what the expert would do. The other baselines that do not use expert demonstrations to guide exploration – RLPD, RFT, and IQL – take longer to get equivalent data in the replay buffer and thus require more steps of exploration to get to a given level of performance.

5.2 Comparison to IBRL

Figure 4 shows a comparison of the learning curves for DGN (using a policy with dropout of 0.5 in the actor MLP to match the architecture of the policy IBRL uses) and IBRL (Hu et al., 2024). These results show that when using a high-quality, single-mode expert dataset (specifically 50 of the "ph" demos provided by Mandlekar et al. (2021)), IBRL slightly outperforms DGN early on in the learning process.

However, the results also show that IBRL is more sensititive to the quality and multimodality of the dataset of expert demonstrations than DGN. In Figure 4, DGN is compared to IBRL when IBRL is using an underfit BC reference policy. The figure reveals that DGN also outperforms IBRL

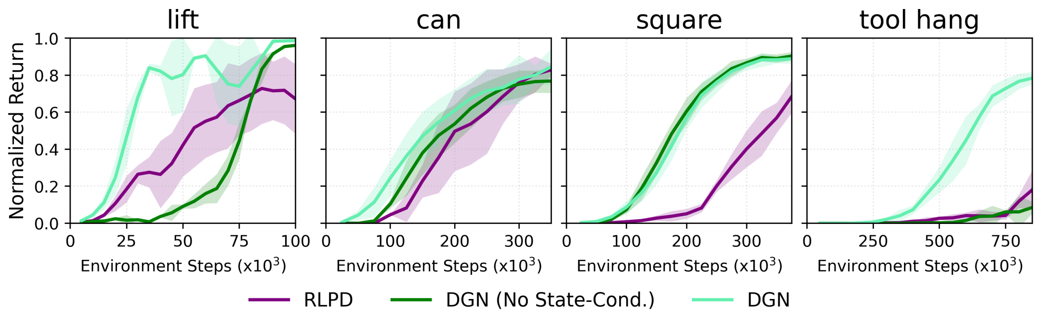

Figure 5: Ablation on state-conditioning the DGN Distribution. Without state-conditioning the learned DGN distribution, the ablation of DGN becomes much less effective at improving exploration, leading to worse sample efficiency than DGN.

when suboptimal hyperparameters are used to train the IBRL reference policy. Not only does this show IBRL to be less robust but also that IBRL likely requires more experimentation to get results equivalent to DGN's. Using IBRL requires two sets of hyperparameter tuning: tuning of the BC policy learning process *and* tuning of the RL process using that reference policy. Oftentimes, producing a good BC policy will require many episodes of evaluation to chose a good stopping checkpoint or tune other hyperparameters. Depending on the task, the number of rollouts done in this process may be similar to or exceed the number of rollouts done for data collection during the RL process itself.

Figure 4 also show that when using 50 successful human demonstrations that are of lower quality and multimodal (specifically "worse" demos from the "mh" subset of expert demos provided by Mandlekar et al. (2021)), DGN again outperforms IBRL. Both these results and the comparison to IBRL with an underfit policy show that DGN is more robust than IBRL. This is likely the case because DGN only needs to guide exploration by approximately following a rough model of the noise distribution, whereas IBRL follows actions from the BC reference policy precisely. IBRL is thus more sensitive to BC policy quality, since the reference policy may simply suggest poor actions when the policy does not model the expert demos well. In contrast, DGN models a distribution of actions (really, action differences). It is more likely that "good" actions to take during exploration will fall somewhere within this DGN distribution than it is that the IBRL BC reference policy will hit upon these "good" actions fairly precisely. Moreover, when the DGN distribution has trouble modeling the expert dataset, the distribution will broaden, suggesting more wide-ranging exploration. In contrast, the IBRL reference policy may simply produce incorrect actions in these cases.

5.3 Ablation on State-Conditioning

To test the importance of state-conditioning the learned DGN distribution, we test an ablation of DGN in which the learned distribution is not state-conditioned. The only change made to DGN for this ablation is that, rather than learning a neural network that maps from states to the entries of (the decomposition of) a covariance matrix, the entries of the entries of (the decomposition of) the one, global covariance matrix are learned directly as parameters.

In Figure 5, this ablation is compared to DGN and RLPD. On most tasks, the ablation of DGN without state-conditioning underperforms DGN, often substantially. As argued in section 3.2, the direction of the errors made by the RL policy will likely differ substantially in different states, so state-conditioning the DGN distribution is necessary to carefully chose the direction in which to explore in any given state.

5.4 Ablation on Number of Demos Used

Figure 6 compares the performance of DGN and RLPD as the number of expert demonstrations provided increases. The performance of both methods improves as the number of expert demonstrations used increases. Using more expert demonstrations in the replay buffers broadens the state coverage of the expert data sampled from the replay buffer during actor-critic updates, improving performance for both RLPD and DGN. However, on the more difficult square task, adding more data improves

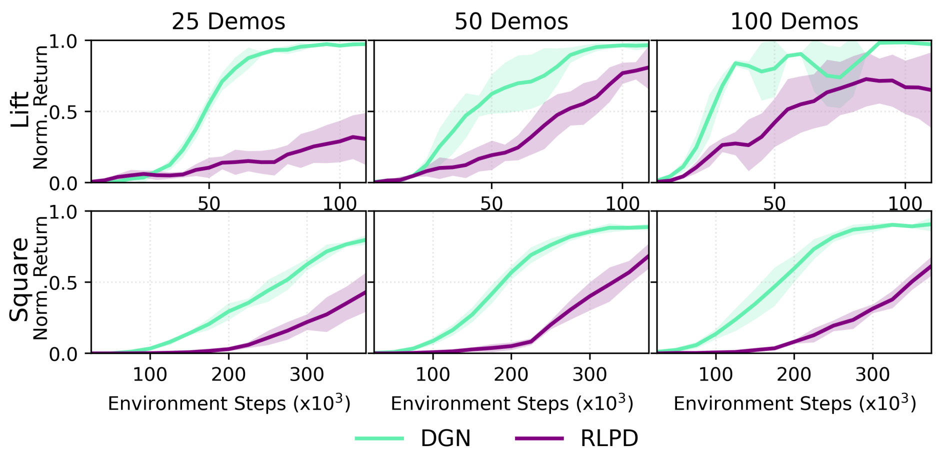

Figure 6: Comparison of the performance of DGN and RLPD with different numbers of demonstrations. Both methods generally improve in performance as the number of demonstrations provided increases, but DGN outperforms RLPD across all tested dataset sizes.

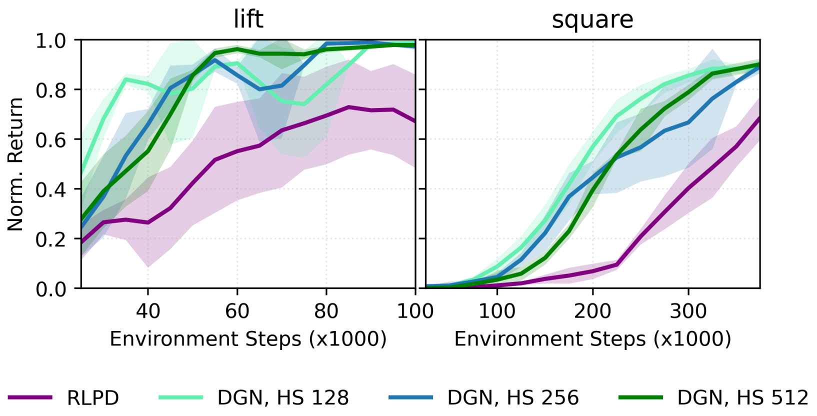

Figure 7: Comparison of the performance of DGN when varying the size of the model underlying the learned DGN noise distribution. DGN is robust to different choices of the hidden size of this MLP.

the performance of DGN more than that of RLPD. On this difficult task, the improved precision of the learned DGN distribution – and thus the effect DGN has on improved targeting of exploration – seems to have a stronger effect on improving sample efficiency than adding those same demos to the replay buffer. If this trend continues with more data on harder tasks, it is a good sign for DGN's ability to scale well.

5.5 Ablation on Size of DGN Model

Figure 7 shows the performance of DGN as the hidden size of the DGN distribution MLP (that maps states to covariance matrices) is varied. For all tested model sizes on both tested tasks, DGN outperforms RLPD. This shows that DGN is not overly sensitive to changes in the architecture of the learned DGN distribution, which bolsters the case for DGN's practical usefulness.

6 Discussion

Potential Broader Impact: Promisingly, the performance of DGN relative to baselines improves on more difficult tasks that are longer-horizon and require more precision. Especially if this trend continues in future work, DGN and derivatives of it may become an essential part of the toolkit of anyone who wants to efficiently bootstrap online RL in difficult, real-world generalist robot tasks. DGN may also be used with a wide variety of off-policy, online RL algorithms, meaning that could be a component of a wide array of other pipelines.

Limitations: A key limitation of DGN is that it primarily improves sample efficiency early on in the RL process. While it would be desirable to have a method that improves sample efficiency throughout the RL process, sample efficiency in the early portions of a RL training run are especially critic because the robot may damage itself when using a poor RL policy that has not been trained much yet.

Another potential limitation is that DGN may have poor OOD (out-of distribution) behavior relative to RLPD and other methods that use expert demonstrations in more subtle ways. For example, if the initial state space during RL training is broader than the initial state space in the demo set, states highly OOD from the demo set might be encountered. The DGN distribution may guide exploration poorly at these OOD states.

Project Difficulties: There are many hyperparameters that go into the algorithm. In general, it is hard to be certain there isn't some weird interplay between two seemingly unrelated hyperparameter choices that could impact the conclusions drawn from results. I learned about using your best judgment to identify what is most important to ablate over.

7 Conclusion

The experiments presented in this report show that *the sample efficiency of DGN is better than that of several widely-used methods that use expert demonstrations, but not to guide exploration*. DGN matches the performance of IBRL (Hu et al., 2024), a state-of-the-art method to improve RL sample efficiency using a BC reference policy, under optimal conditions. However, *DGN is shown to be more robust than IBRL*, outperforming it in potentially more realistic situations with suboptimal data quality, dataset multimodality, and suboptimally trained reference policies.

By improving the sample efficiency of online RL, DGN and related methods may help make expert-demo-aided RL competitive with imitation-based methods that currently dominate the frontier of robot policy learning.

There are many other directions for future work, including:

- Trying DGN with image-based observations. This raises several nontrivial questions, such as how to train the image encoder for the DGN distribution model. My coauthors and I plan to do this soon, including on real-world tasks.
- Improving OOD performance of DGN by training the learned distribution to be a standard normal on OOD states (Perhaps by regularizing the DGN distribution to a standard normal at randomly chosen states alongside the current learning objective).
- Investigating better motivated metrics for shutting off DGN or experimenting with "soft" forms of shutting off DGN.
- Investigating more architectures for the DGN distribution. (Perhaps sample action chunks of noise rather than noise for a single action. Perhaps use quantized CVAEs or strongly regularized diffusion models for the distribution.)
- Investigate using a modified form of DGN that can work with diffusion policies and other policy classes. The current form of DGN relies heavily on the RL policy being deterministic (or, equivalently, the mean of a Gaussian). Perhaps the core idea of modelling the distribution of RL action and expert action differences can be extended to other policy classes?

I am excited to see what will come of these future extensions, variants, and implementations of DGN.

8 Team Contributions

The work presented in this report was done as part of a larger project with Annie Chen, Perry Dong, and Chelsea Finn. I am co-first author on this work. With the exception of one figure that my co-author Annie Chen made very well and gave me permission to use here, the writing in this report is in my own words, done specifically for this report. In this report, I also do not include some portions of that project for which I am less directly responsible, such as results on another suite of tasks where one of my co-authors did much of the implementation of those experiments.

Changes from Proposal In the proposal I was planning to try this algorithm on top of IBRL. Instead this report focused on implementing it on top of RLPD.

References

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. 2018. Hindsight Experience Replay. arXiv:1707.01495 [cs.LG] `https://arxiv.org/abs/1707.01495`

Philip J. Ball, Laura Smith, Ilya Kostrikov, and Sergey Levine. 2023. Efficient Online Reinforcement Learning with Offline Data. arXiv:2302.02948 [cs.LG] `https://arxiv.org/abs/2302.02948`

Scott Fujimoto, Herke van Hoof, and David Meger. 2018. Addressing Function Approximation Error in Actor-Critic Methods. arXiv:1802.09477 [cs.AI] `https://arxiv.org/abs/1802.09477`

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. arXiv:1801.01290 [cs.LG] `https://arxiv.org/abs/1801.01290`

Hengyuan Hu, Suvir Mirchandani, and Dorsa Sadigh. 2024. Imitation Bootstrapped Reinforcement Learning. `https://openreview.net/forum?id=Ap344YqCcD`

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. 2021. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169* (2021).

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2019. Continuous control with deep reinforcement learning. arXiv:1509.02971 [cs.LG] `https://arxiv.org/abs/1509.02971`

Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. 2021. What Matters in Learning from Offline Human Demonstrations for Robot Manipulation. arXiv:2108.03298 [cs.RO] `https://arxiv.org/abs/2108.03298`

Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. 2020. Awac: Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359* (2020).

Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. 2018. Overcoming Exploration in Reinforcement Learning with Demonstrations. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*. 6292–6299. `https://doi.org/10.1109/ICRA.2018.8463162`

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. 2019. Advantage-Weighted Regression: Simple and Scalable Off-Policy Reinforcement Learning. arXiv:1910.00177 [cs.LG] `https://arxiv.org/abs/1910.00177`

Mel Vecerik, Todd Hester, Jonathan Scholz, Fumin Wang, Olivier Pietquin, Bilal Piot, Nicolas Heess, Thomas Rothörl, Thomas Lampe, and Martin Riedmiller. 2018. Leveraging Demonstrations for Deep Reinforcement Learning on Robotics Problems with Sparse Rewards. arXiv:1707.08817 [cs.AI] `https://arxiv.org/abs/1707.08817`