Continuous Deep Q-Learning with Model-based Acceleration: Appendix

Shixiang Gu[1 2 3] SG717@CAM.AC.UK
Timothy Lillicrap[4] COUNTZERO@GOOGLE.COM
Ilya Sutskever[3] ILYASU@GOOGLE.COM
Sergey Levine[3] SLEVINE@GOOGLE.COM
[1]University of Cambridge [2]Max Planck Institute for Intelligent Systems [3]Google Brain [4]Google DeepMind

1. Appendix

1.1. iLQG

The iLQG algorithm optimizes trajectories by iteratively constructing locally optimal linear feedback controllers under a local linearization of the dynamics $p(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t, \boldsymbol{u}_t) = \mathcal{N}(\boldsymbol{f}_{\boldsymbol{x}t}\boldsymbol{x}_t + \boldsymbol{f}_{\boldsymbol{u}t}\boldsymbol{u}_t, \boldsymbol{F}_t)$ and a quadratic expansion of the rewards $r(\boldsymbol{x}_t, \boldsymbol{u}_t)$ (Tassa et al., 2012). Under linear dynamics and quadratic rewards, the action-value function $Q(\boldsymbol{x}_t, \boldsymbol{u}_t)$ and value function $V(\boldsymbol{x}_t)$ are locally quadratic and can be computed by dynamics programming.[1]

$$\begin{aligned} Q_{\boldsymbol{x}\boldsymbol{u},\boldsymbol{x}\boldsymbol{u}t} &= r_{\boldsymbol{x}\boldsymbol{u},\boldsymbol{x}\boldsymbol{u}t} + \boldsymbol{f}_{\boldsymbol{x}\boldsymbol{u}t}^T V_{\boldsymbol{x},\boldsymbol{x}t+1}\boldsymbol{f}_{\boldsymbol{x}\boldsymbol{u}t} \\ Q_{\boldsymbol{x}\boldsymbol{u}t} &= r_{\boldsymbol{x}\boldsymbol{u}t} + \boldsymbol{f}_{\boldsymbol{x}\boldsymbol{u}t}^T V_{\boldsymbol{x},\boldsymbol{x}t+1} \\ V_{\boldsymbol{x},\boldsymbol{x}t} &= Q_{\boldsymbol{x},\boldsymbol{x}t} - Q_{\boldsymbol{u},\boldsymbol{x}t}^T Q_{\boldsymbol{u},\boldsymbol{u}t}^{-1} Q_{\boldsymbol{u},\boldsymbol{x}t} \\ V_{\boldsymbol{x}t} &= Q_{\boldsymbol{x}t} - Q_{\boldsymbol{u},\boldsymbol{x}t}^T Q_{\boldsymbol{u},\boldsymbol{u}t}^{-1} Q_{\boldsymbol{u}t} \\ Q_{\boldsymbol{x},\boldsymbol{x}T} &= V_{\boldsymbol{x},\boldsymbol{x}T} = r_{\boldsymbol{x},\boldsymbol{x}T} \end{aligned} \tag{1}$$

The time-varying linear feedback controller $\boldsymbol{g}(\boldsymbol{x}_t) = \hat{\boldsymbol{u}}_t + \boldsymbol{k}_t + \boldsymbol{K}_t(\boldsymbol{x}_t - \hat{\boldsymbol{x}}_t)$ maximizes the locally quadratic Q, where $\boldsymbol{k}_t = -Q_{\boldsymbol{u},\boldsymbol{u}t}^{-1} Q_{\boldsymbol{u}t}$, $\boldsymbol{K}_t = -Q_{\boldsymbol{u},\boldsymbol{u}t}^{-1} Q_{\boldsymbol{u},\boldsymbol{x}t}$, and $\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{u}}_t$ denote states and actions of the current trajectory around which the partial derivatives are computed. Employing the maximum entropy objective (Levine & Koltun, 2013), we can also construct a linear-Gaussian controller, where c is a scalar to adjust for arbitrary scaling of the reward magnitudes,

$$\pi_t^{iLQG}(\boldsymbol{u}_t|\boldsymbol{x}_t) = \mathcal{N}(\hat{\boldsymbol{u}}_t + \boldsymbol{k}_t + \boldsymbol{K}_t(\boldsymbol{x}_t - \hat{\boldsymbol{x}}_t), -cQ_{\boldsymbol{u},\boldsymbol{u}t}^{-1}) \tag{2}$$

When the dynamics are not known, a particularly effective way to use iLQG is to combine it with learned time-varying linear models $\hat{p}(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t, \boldsymbol{u}_t)$. In this variant of the algorithm, trajectories are sampled from the controller in Equation (2) and used to fit time-varying linear dynamics with linear regression. These dynamics are then used with iLQG to obtain a new controller, typically using a KL-divergence constraint to enforce a trust region, so that the new controller doesn't deviate too much from the region in which the samples were generated (Levine & Abbeel, 2014).

[1]While standard iLQG notation denotes Q, V as discounted sum of costs, we denote them as sum of rewards to make them consistent with the rest of the paper

1.2. Locally-Invariant Exploration for Normalized Advantage Functions

Exploration is an essential component of reinforcement learning algorithms. The simplest and most common type of exploration involves randomizing the actions according to some distribution, either by taking random actions with some probability (Mnih et al., 2015), or adding Gaussian noise in continuous action spaces (Schulman et al., 2015). However, choosing the magnitude of the random exploration noise can be difficult, particularly in high-dimensional domains where different action dimensions require very different exploration scales. Furthermore, independent (spherical) Gaussian noise may be inappropriate for tasks where the optimal behavior requires correlation between action dimensions, as for example in the case of the swimming snake described in our experiments, which must coordinate the motion of different body joints to produce a synchronized undulating gait.

The NAF provides us with a simple and natural avenue to obtain an adaptive exploration strategy, analogously to Boltzmann exploration. The idea is to use the matrix in the quadratic component of the advantage function as the precision for a Gaussian action distribution. This naturally causes the policy to become more deterministic along directions where the advantage function varies steeply, and more random along directions where it is flat. The corresponding policy is given by

$$\begin{aligned} \pi(\boldsymbol{u}|\boldsymbol{x}) &= \exp^{Q(\boldsymbol{x},\boldsymbol{u}|\theta^Q)} / \int \exp^{Q(\boldsymbol{x},\boldsymbol{u}|\theta^Q)} d\boldsymbol{u} \\ &= \mathcal{N}(\boldsymbol{\mu}(\boldsymbol{x}|\theta^\mu), c\boldsymbol{P}(\boldsymbol{x}|\theta^P)^{-1}). \end{aligned} \tag{3}$$

Previous work also noted that generating Gaussian exploration noise independently for each time step was not well-

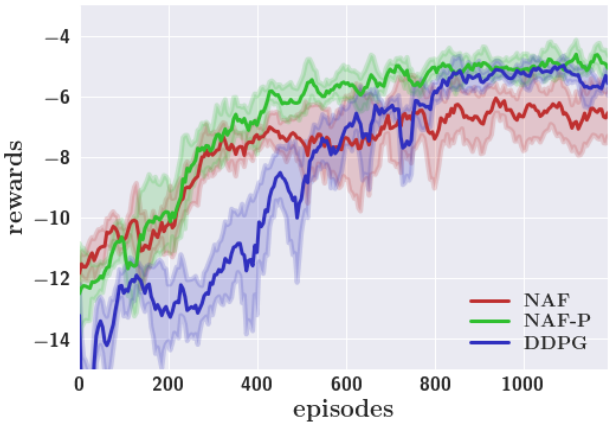


Figure 1. NAF with exploration noise generated using the precision term (NAF-P) slightly outperforms the best DDPG result. Precision term is not used until episode 200.

suited for many continuous control tasks, particularly simulated robotic tasks where the actions correspond to torques or velocities (Lillicrap et al., 2016). The intuition is that, as the length of the time-step decreases, temporally independent Gaussian exploration will cancel out between time steps. Instead, prior work proposed to sample noise from an Ornstein-Uhlenbech (OU) process to generate a temporally correlated noise sequence (Lillicrap et al., 2016). We adopt the same approach in our work, but sample the innovations for the OU process from the Gaussian distribution in Equation 3. Lastly, we note that the overall scale of $\boldsymbol{P}(\boldsymbol{x}|\theta^P)$ could vary significantly through the learning, and depends on the magnitude of the cost, which introduces an undesirable additional degree of freedom. We therefore use a heuristic adaptive-scaling trick to stabilize the noise magnitudes.

Using the learned precision as the noise covariance for exploration allowed for convergence to a better policy on the "canada2d" task, which requires using an arm to strike a puck toward a target, as shown in Figure 1, but did not make a significant difference on the other domains.

1.3. Descriptions of Task Domains

Table 1 describes the task domains used in the experiments.

1.4. More Results on Normalized Advantage Functions

Figures 2a, 2b, and 2c provide additional results on the comparison experiments between DDPG and NAF. As shown in the main paper, NAF generally outperforms DDPG. In certain tasks that require precision, such as peg insertion, the difference is very noticeable. However, there are also few cases where NAF underperforms DDPG. The most consistent of such cases is cheetah. While both DDPG and NAF enable cheetah to run decent distances, it is often observed that the cheetah movements learned in NAF are little less natural than those from DDPG. We speculate such behaviors come from the uni-modal behavior, and thus exploring other parametric forms of NAF, such as multi-modal variants, is a promising avenue for future work.

1.5. More Results on Evaluating Best-Case Model-Based Improvement with True Models

Domains	-	0.5	ImR	ImR,0.5	ImR,1
Reacher	-0.488	-0.449	-0.448	**-0.426**	-0.548
episodes	740	670	450	430	**90**
Canada2d	-6.23	-6.23	-5.89	**-5.88**	-12.0
episodes	1970	1580	570	**140**	210
Cheetah	7.00	7.10	**7.36**	7.29	6.43
episodes	580	1080	590	740	**390**

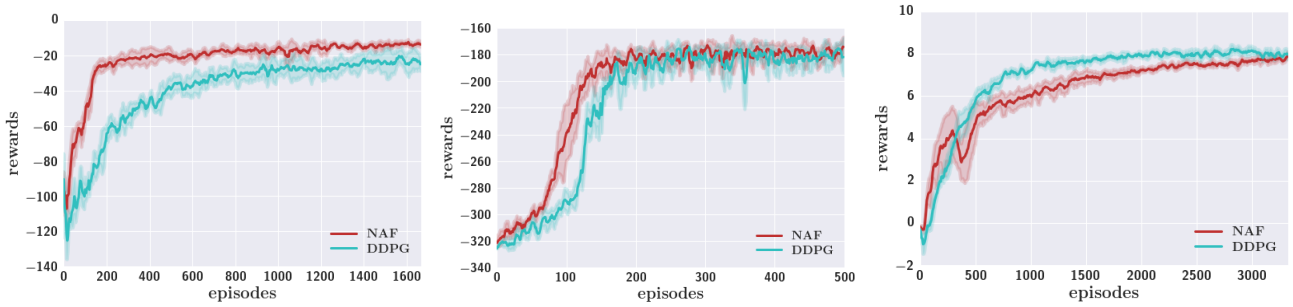
Table 2. Best-case model-based acceleration with true dynamics models. Best test rewards of NAF policies (first row), and the episodes it required to reach 5% of the best value (second row). "0.5" and "1" correspond to the fraction of MPC episodes. "ImR" means using imagination rollout with rollout length $l = 10$ for reacher, canada2d, and $l = 5$ for cheetah.

In the main paper, iLQG with true dynamics is used to generate guided exploration trajectories. While iLQG works for simple manipulation tasks with small number of initial states, it does not work well for random target reacher or complex locomotion tasks such as cheetah. We therefore run iLQG in model-predictive control (MPC) mode for the experiments reported in Figures 3c, 3b, and 3a, and Table 2. It is important to note that for those experiments, the hyper-parameters were fixed (batch normalization is on, learning rate is 10^{-3}, and exploration size is 0.3), and thus the results differ slightly from the experiments in the previous section.

In cheetah and other complex locomotion tasks, MPC policy is usually sub-optimal, and thus poor performance of mixing MPC experience in Figure 3b is expected. On the other hand, MPC policy works reasonably in hard manipulation tasks such as canada2d, and there is significant gain from mixing MPC experience as Figure 3c shows. However, the most consistent gain comes from using imagination rollouts. In particular, Figure 3c shows that in canada2d, MPC experiences gives very good trajectories, i.e. those that hit balls in roughly the right directions, and doing rollouts can generate more of this useful experience, enabling canada2d to learn very quickly. While with true dynamics having the imagination experience directly means more experience and such result may be trivial, it is important to see the benefits of rollouts which only explore up to $l = 10$ steps away from the real trajectories. This means the dynamics model only needs to be accurate around the data trajectories and this significantly lessens the requirement on fitted models.

Domain	Description	Domain	Description
Cartpole	The classic cart-pole swing-up task. Agent must balance a pole attached to a cart by applying forces to the cart alone. The pole starts each episode hanging upside-down.	Reacher	Agent is required to move a 3-DOF arm from random starting locations to random target positions.
Peg	Agent is required to insert the tip of a 3-DOF arm from locally-perturbed starting locations to a fixed hole.	Gripper	Agent must use an arm with gripper appendage to grasp an object and manuver the object to a fixed target.
GripperM	Agent must use an arm with gripper attached to a moveable platform to grasp an object and move it to a fixed target.	Canada2d	Agent is required to use an arm with hockey-stick like appendage to hit a ball initialzed to a random start location to a random target location.
Cheetah	Agent should move forward as quickly as possible with a cheetah- like body that is constrained to the plane.	Swimmer6	Agent should swim in snake-like manner toward the fixed target using six joints, starting from random poses.
Ant	The four-legged ant should move toward the fixed target from a fixed starting position and posture.	Walker2d	Agent should move forward as quickly as possible with a bipedal walker constrained to the plane without falling down or pitching the torso too far forward or backward.

Table 1. List of domains. All the domains except ant are 2D.

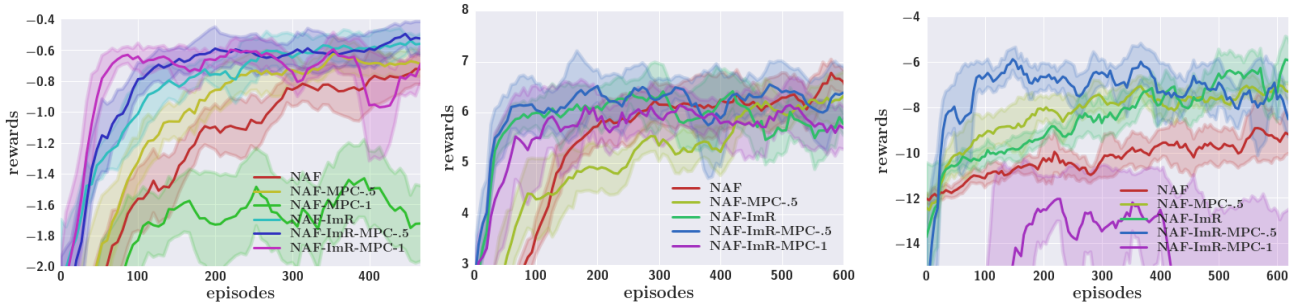


(a) NAF significantly outperforms DDPG on moving gripper.

(b) NAF converges faster than DDPG on swimmer6.

(c) DDPG converges faster than NAF on cheetah.

Figure 2. NAF vs DDPG on three domains.



(a) NAF on multi-target reacher. Insignificant gain from mixing MPC experience. Significant gain from imagination rollouts.

(b) NAF on cheetah. Great speeds up with imagination rollouts, no gain from mixing MPC experiences.

(c) NAF on canada2d. Very significant speed-ups from mixing MPC experiences, both with or without the rollouts.

Figure 3. NAF on multi-target reacher, cheetah, and canada2d, with model-based acceleration using true dynamics: "ImR" denotes using the imagination rollout, $l = 10$ steps. "MPC-x" indicates mixing x fraction of MPC episodes.

References

Levine, Sergey and Abbeel, Pieter. Learning neural network policies with guided policy search under unknown dynamics. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 1071–1079, 2014.

Levine, Sergey and Koltun, Vladlen. Guided policy search. In *International Conference on Machine Learning (ICML)*, pp. 1–9, 2013.

Lillicrap, Timothy P, Hunt, Jonathan J, Pritzel, Alexander, Heess, Nicolas, Erez, Tom, Tassa, Yuval, Silver, David, and Wierstra, Daan. Continuous control with deep reinforcement learning. *International Conference on Learning Representations (ICLR)*, 2016.

Mnih, Volodymyr, Kavukcuoglu, Koray, Silver, David, Rusu, Andrei A, Veness, Joel, Bellemare, Marc G, Graves, Alex, Riedmiller, Martin, Fidjeland, Andreas K, Ostrovski, Georg, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

Schulman, John, Levine, Sergey, Abbeel, Pieter, Jordan, Michael I., and Moritz, Philipp. Trust region policy optimization. In *International Conference on Machine Learning (ICML)*, pp. 1889–1897, 2015.

Tassa, Yuval, Erez, Tom, and Todorov, Emanuel. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *International Conference on Intelligent Robots and Systems (IROS)*, pp. 4906–4913. IEEE, 2012.