VARIANCE REDUCTION FOR REINFORCEMENT LEARNING IN INPUT-DRIVEN ENVIRONMENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

We consider reinforcement learning in input-driven environments, where an exogenous, stochastic input process affects the dynamics of the system. Input processes arise in many applications, including queuing systems, robotics control with disturbances, and object tracking. Since the state dynamics and rewards depend on the input process, the state alone provides limited information for the expected future returns. Therefore, policy gradient methods with standard state-dependent baselines suffer high variance during training. We derive a bias-free, input-dependent baseline to reduce this variance, and analytically show its benefits over state-dependent baselines. We then propose a meta-learning approach to overcome the complexity of learning a baseline that depends on a long sequence of inputs. Our experimental results show that across environments from queuing systems, computer networks, and MuJoCo robotic locomotion, input-dependent baselines consistently improve training stability and result in better eventual policies.

1 INTRODUCTION

Deep reinforcement learning (RL) has emerged as a powerful approach to sequential decision-making problems, achieving impressive results in domains such as game playing (Mnih et al., 2015; Silver et al., 2017), robotics (Levine et al., 2016), and continuous control (Schulman et al., 2015a; Lillicrap et al., 2015). This paper concerns RL in *input-driven environments*. Informally, input-driven environments have dynamics that are partially dictated by an exogenous, stochastic *input process*. Queuing systems (Kleinrock, 1976; Kelly, 2011) are an example; their dynamics is governed by not only the decisions made within the system (e.g., scheduling, load balancing) but also the arrival process that brings work (e.g., jobs, customers, packets) into the system. Input-driven environments also arise naturally in many other domains: network control and optimization (Winstein & Balakrishnan, 2013; Mao et al., 2017), robotics control with stochastic disturbances (Pinto et al., 2017), locomotion in environments with complex terrains and obstacles (Heess et al., 2017), tracking moving targets, and more (see Figure 1).

We focus on model-free policy gradient RL algorithms (Williams, 1992; Mnih et al., 2016; Schulman et al., 2015a), which have been widely adopted and benchmarked for a variety of RL tasks (Duan et al., 2016; Wu & Tian, 2017). Policy gradient algorithms optimize the policy parameters by estimating the gradient of the expected total reward (or "return") using Monte Carlo techniques (Owen, 2013). An important challenge for these methods is high variance in the gradient estimates, as it increases sample complexity and can impede effective learning altogether when training non-linear neural network policies (Schulman et al., 2015b; Mnih et al., 2016). A standard approach to reduce variance is to subtract a "baseline" from the total reward to estimate the gradient (Weaver & Tao, 2001). The baseline is usually a function of the state. The most common choice is the *value function* — the expected return starting from the state. The interpretation of this baseline is to compare the return for an action taken in a particular state to the average return achieved from that state, and increase or decrease the probability of the action based on whether its return is better or worse than average.

Our main insight is that a state-dependent baseline — such as the value function — is a poor choice in input-driven environments, whose state dynamics and rewards are partially driven by the input process. In such environments, comparing the return to the value-function baseline may provide limited information about the quality of actions. A strong action could end up with a lower-than-average return if the input sequence following the action is unfavorable; similarly, a poor action might achieve a good return with an advantageous input sequence. Intuitively, a good baseline for policy

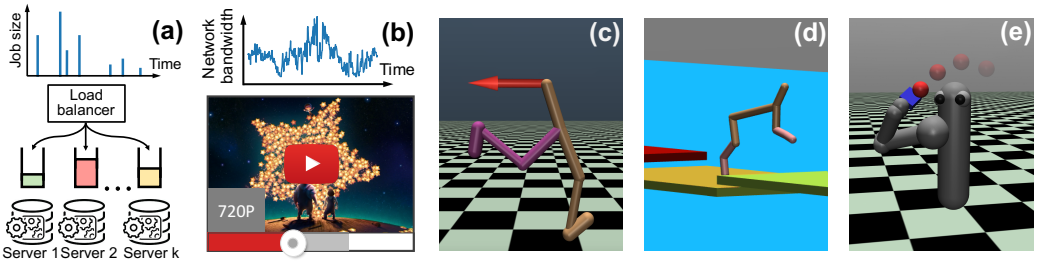

Figure 1: Input-driven environments: *(a)* load-balancing heterogeneous servers (Harchol-Balter & Vesilo, 2010) with stochastic job arrival as input process; *(b)* adaptive bitrate video streaming (Mao et al., 2017) with stochastic network bandwidth as input process; *(c)* Walker2d in wind with a stochastic force (wind) applied to the walker as input; *(d)* HalfCheetah on floating tiles with the stochastic process that controls the buoyancy of the tiles as input; *(e)* 7-DoF arm tracking moving target with the stochastic target position as input. Environments *(c)–(e)* use the MuJoCo (Todorov et al., 2012) physics simulator.

gradient estimation should take the specific instance of the input process — the sequence of input values — into account. We call such a baseline an *input-dependent baseline*; it is a function of both the state and the entire future input sequence.

We formally define input-driven Markov decision processes, and we prove that an input-dependent baseline does not introduce bias in standard policy gradient algorithms such as Advantage Actor Critic (A2C) (Mnih et al., 2016) and Trust Policy Region Optimization (TRPO) (Schulman et al., 2015a) provided that the input process is independent of the states and actions. We derive the optimal input-independent baseline and a simpler one to work with in practice; this takes the form of a *conditional value function* — the expected return given the state and the future input sequence.

Input-dependent baselines are harder to learn than their state-dependent counterparts; they are high-dimensional functions of the sequence of input values. To learn input-dependent baselines efficiently, we propose a simple approach based on meta learning (Finn et al., 2017; Vilalta & Drissi, 2002). The idea is to learn a "meta baseline" that can be specialized to a baseline for a specific input instantiation using a small number of training episodes with that input. This approach can be used in applications in which we can repeat an input sequence multiple times during training, such as applications using simulations or experiments with previously-collected input traces for training (McGough et al., 2017).

We compare our input-dependent baseline to the standard value function baseline for the five tasks illustrated in Figure 1. These tasks are derived from queuing systems (load balancing heterogeneous servers (Harchol-Balter & Vesilo, 2010)), computer networks (bitrate adaptation for video streaming (Mao et al., 2017)), and variants of standard continuous control RL benchmarks in the MuJoCo (Todorov et al., 2012) physics simulator. We adapted three widely-used MuJoCo benchmarks (Duan et al., 2016; Clavera et al., 2018; Heess et al., 2017) to add a stochastic input element that makes these tasks significantly more challenging. For example, we replaced the static target in a 7-DoF robotic arm target-reaching task with a randomly-moving target that the robot aims to track over time. Our results show that input-dependent baselines consistently provide improved training stability and better eventual policies. Input-dependent baselines are applicable to a variety of policy gradient methods, including A2C, TRPO, PPO, and also benefit robust adversarial RL methods such as RARL (Pinto et al., 2017). Videos demonstrations of our experiments are available at `https://sites.google.com/view/input-dependent-baseline/`.

2 Preliminaries

Notation. We consider a discrete-time Markov decision process (MDP), defined by $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \rho_0, r, \gamma)$, where $\mathcal{S} \subseteq \mathbb{R}^n$ is a set of n-dimensional states, $\mathcal{A} \subseteq \mathbb{R}^m$ is a set of m-dimensional actions, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the state transition probability distribution, $\rho_0 : \mathcal{S} \to [0, 1]$ is the distribution over initial states, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, and $\gamma \in (0, 1)$ is the discount factor. We denote a stochastic policy as $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$, which aims to optimize the expected return $\eta(\pi) = \mathbb{E}_\tau \left[\sum_{t=0}^{\infty} r(s_t, a_t)\right]$, where $\tau = (s_0, a_0, ...)$ is the trajectory following $s_0 \sim \rho_0$, $a_t \sim \pi(a_t|s_t)$, $s_{t+1} \sim \mathcal{P}(s_{t+1}|s_t, a_t)$. We use $V_\pi(s_t) = \mathbb{E}_{a_t, s_{t+1}, a_{t+1}, ...} \left[\sum_{l=0}^{\infty} \gamma^l r(s_{t+l}, a_{r+l})\right]$ to define the value function, and $Q_\pi(s_t, a_t) = \mathbb{E}_{s_{t+1}, a_{t+1}, ...} \left[\sum_{l=0}^{\infty} \gamma^l r(s_{t+l}, a_{r+l})\right]$ to define the

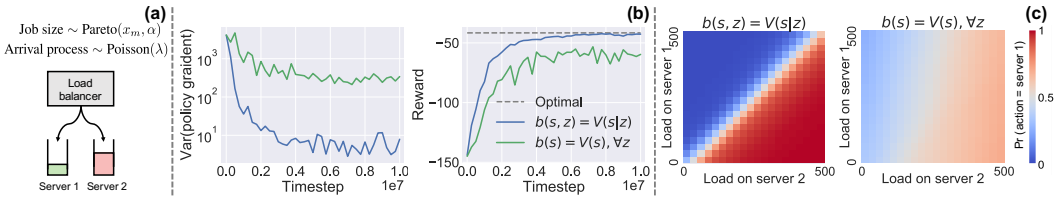

Figure 2: Load balancing over two servers. *(a)* Job sizes follow a Pareto distribution and jobs arrive as a Poisson process; the RL agent observes the queue lengths and picks a server for an incoming job. *(b)* The input-dependent baseline (blue) results in a 50× lower policy gradient variance (left) and a 33% higher test reward (right) than the standard, state-dependent baseline (green). *(c)* The probability heatmap of picking server 1 shows that using the input-dependent baseline (left) yields a more precise policy than using the state-dependent baseline (right).

state-action value function. For any sequence $(x_0, x_1, ...)$, we use $\boldsymbol{x}$ to denote the entire sequence and $x_{i:j}$ to denote $(x_i, x_{i+1}, ..., x_j)$.

Policy Gradient Methods. Policy gradient methods estimate the gradient of expected return with respect to the policy parameters (Sutton et al., 1999; Kakade, 2002; Gu et al., 2017). To train a policy π_θ parameterized by θ, the Policy Gradient Theorem (Sutton et al., 1999) states that

$$\nabla_\theta \eta(\pi_\theta) = \mathbb{E}_{\substack{s\sim\rho_\pi \\ a\sim\pi_\theta}} \left[\nabla_\theta \log \pi_\theta(a|s) Q_{\pi_\theta}(s,a)\right], \tag{1}$$

where $\rho_\pi(s) = \sum_{t=0}^{\infty} \left[\gamma^t \Pr(s_t = s)\right]$ denotes the discounted state visitation frequency. Practical algorithms often use the undiscounted state visitation frequency (i.e., $\gamma = 1$ in ρ_π), which can make the estimation slightly biased (Thomas, 2014).

Estimating the policy gradient using Monte Carlo estimation for the Q function suffers from high variance (Mnih et al., 2016). To reduce variance, an appropriately chosen baseline $b(s_t)$ can be subtracted from the Q-estimate without introducing bias (Greensmith et al., 2004). The policy gradient estimation with a baseline in Eq. (1) becomes $\mathbb{E}_{\rho_\pi,\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a|s)\left(Q_{\pi_\theta}(s,a) - b(s)\right)\right]$. While an optimal baseline exists (Greensmith et al., 2004; Wu et al., 2018), it is hard to estimate and often replaced by the value function $b(s_t) = V_\pi(s_t)$ (Sutton & Barto, 1998; Mnih et al., 2016).

Stochastic gradient descent using Eq. (1) does not guarantee consistent policy improvement in complex control problems. Trust Region Policy Optimization (TRPO) (Schulman et al., 2015a) is an alternative approach that offers monotonic policy improvements. TRPO maximizes a surrogate objective, subject to a KL divergence constraint:

$$\underset{\theta}{\text{maximize}} \quad \mathbb{E}_{\substack{s\sim\rho_{\pi_{\text{old}}} \\ a\sim\pi_{\text{old}}}} \left[\frac{\pi_\theta(a|s)}{\pi_{\text{old}}(a|s)} Q_{\pi_{\text{old}}}(s,a)\right] \tag{2}$$

$$\text{subject to} \quad \mathbb{E}_{s\sim\rho_{\pi_{\text{old}}}}\left[D_{\text{KL}}\left(\pi_{\text{old}}(\cdot|s) || \pi_\theta(\cdot|s)\right)\right] \le \delta, \tag{3}$$

in which δ serves as a step size for policy update. Using a baseline in the TRPO objective, i.e. replacing $Q_{\pi_{\text{old}}}(s,a)$ with $Q_{\pi_{\text{old}}}(s,a) - b(s)$, empirically improves policy performance (Schulman et al., 2015b).

3 Motivating Example

We illustrate the variance introduced into policy gradient methods by an exogenous input process using a simple load balancing example (Figure 2a). Jobs arrive over time and must be sent to one of two servers. The state $s_t = (q_1, q_2)$ denotes the current queue lengths at the two servers. The action $a_t \in \{1, 2\}$ picks which server to enqueue the incoming job at. To minimize average job completion time, the reward r_t is $-\tau \times j$, where τ is the time elapsed since last action and j is number of enqueued jobs. The servers process jobs at identical rates, the job sizes follow a Pareto distribution (scale $x_m = 100$, shape $\alpha = 1.5$), and jobs arrive in a Poisson process ($\lambda = 55$). Intuitively, the optimal policy for this simple example is to join the shortest queue (Daley, 1987). The optimal policy for more general versions of the load balancing problem (e.g., with heterogeneous processing rates) is not known (Harchol-Balter & Vesilo, 2010); we evaluate such cases in §6.2.

Since the Pareto distribution is heavy-tailed, the queue occupancies and the return over a long time horizon have large variance. We train two A2C agents (Mnih et al., 2016; Dhariwal et al., 2017), one with the standard value function baseline and the other with an input-dependent baseline that is

tailored to each specific instantiation of the job arrival process (the details of this baseline are in §4). Figure 2b shows that the input-dependent baseline significantly reduces the variance of the policy gradient. As a result, the learned policy improves compared to the value function baseline. Figure 2c visualizes the policies learned using the two baselines. The optimal policy (pick-shortest-queue) corresponds to a clear divide between the chosen servers at the diagonal. The policy learned with the input-dependent baseline comes much closer to this ideal than with the standard value function baseline, whose fuzzier probability heatmap indicates an unstable, high-variance policy.

In this example, the value function baseline performs poorly because of the variance caused by the input process. In fact, the variance of the standard baseline can be arbitrarily large: we refer the reader to Appendix A for an analytical example on a 1D grid world.

4 Reducing Variance for Input-Driven MDPs

We now formally define input-driven MDPs and derive variance-reducing baselines for policy gradient methods in environments with input processes.

Definition 1. *An input-driven MDP is defined by* $(\mathcal{S}, \mathcal{A}, \mathcal{Z}, \mathcal{P}_s, \mathcal{P}_z, \rho_0^s, \rho_0^z, r, \gamma)$*, where* $\mathcal{Z} \subseteq \mathbb{R}^k$ *is a set of* k*-dimensional input values,* $\mathcal{P}_s(s_{t+1}|s_t, a_t, z_t)$ *is the transition kernel of the states,* $\mathcal{P}_z(z_{t+1}|z_t)$ *is the transition kernel of the input process,* $\rho_0^z(z_0)$ *is the distribution of the initial input,* $r(s_t, a_t, z_t)$ *is the reward function, and* $\mathcal{S}$*,* $\mathcal{A}$*,* ρ_0^s*,* γ *follow the standard definition in §2.*

An input-driven MDP adds an *input process*, z_t, to a standard MDP. For simplicity, we consider only Markov input processes, where z_t depends only on the previous input value z_{t-1}; generalizing to non-Markov input processes is straightforward. In an input-driven MDP, the next state s_{t+1} depends on (s_t, a_t, z_t). The input process is exogenous in the sense that z_t is independent of the processes s_t and a_t. We seek to learn policies that maximize cumulative expected rewards. We focus on two cases, corresponding to the graphical models shown in Figure 3:
Case 1: z_t is a Markov process, and s_t and z_t are both observed at time t. The action a_t can hence depend on both s_t, z_t.
Case 2: z_t is an i.i.d. process, independent of the states and actions, and is not observed at time t. The action a_t can depend only on s_t.

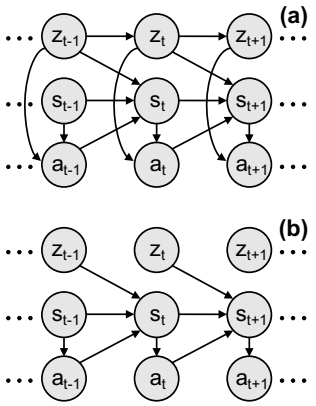

Figure 3: Graphical model of input-driven MDPs. (a) z_t is Markov. (b) z_t is i.i.d.

Proposition 1. *An input-driven MDP satisfying the conditions of either case 1, or case 2 above, is a fully observable MDP.*

Proof. See Appendix B.

Understanding this proposition for case 1 is straightforward. Since both s_t and z_t are observed, considering the tuple (s_t, z_t) to be the 'state' at time t leads trivially to a standard fully observable MDP. On the other hand, with case 2 we see that even if z_t is not observed, we have a fully observable MDP. The intuition for this result comes by viewing z_t as an i.i.d. source of randomness within the state-transition kernel of the MDP.

We now consider policy gradient methods for learning a policy for input-driven MDPs. For the remainder of this paper, we focus on MDPs with the more general structure of case 1 above. Extending our results to case 2 is straightforward.

4.1 Variance Reduction

In input-driven MDPs, the standard input-agnostic baseline is ineffective at reducing variance. We propose to use an *input-dependent* baseline, of the form $b(s_t, z_{t:\infty})$ — a function of both the current state and the specific future input sequence encountered during each training episode. Using this modified baseline is feasible because the future input sequence $z_{t:\infty}$ is known *at training time*. Specifically, following any training episode, we can observe the entire sequence of input values, and use them to compute the baseline for each step t. It is important to note that the policy cannot use the future input values. At time t, the policy only depends only on (s_t, z_t).

We now analyze the effect of using an input-dependent baseline. We show that input-dependent baselines are bias-free, and we derive the optimal input-dependent baseline for variance reduction.

We first state two useful lemmas required for our analysis. The first lemma shows that under the input-driven MDP definition, the input sequence is conditionally independent of the actions, while the second lemma states the input-dependent version of the policy gradient theorem.

In the following, we abuse notation and let s_t denote the joint state (s_t, z_t) which includes the input at time t.

Lemma 1. $\Pr(\boldsymbol{z}, s_t, a_t) = \Pr(\boldsymbol{z}) \Pr(s_t|\boldsymbol{z})\pi_\theta(a_t|s_t)$, *i.e.,* $\boldsymbol{z} - s_t - a_t$ *forms a Markov chain.*

Proof. See Appendix C.

Lemma 2. *For an input-driven MDP, the Policy Gradient Theorem (Eq. (1)) can be rewritten as*

$$\nabla_\theta \eta(\pi_\theta) = \mathbb{E}_{\substack{\boldsymbol{z}\sim P_{\boldsymbol{z}} \\ s\sim\rho_{\pi,\boldsymbol{z}} \\ a\sim\pi_\theta}} \Big[\nabla_\theta \log \pi_\theta(a|s) Q(s, a|\boldsymbol{z})\Big], \tag{4}$$

where $\rho_{\pi,\boldsymbol{z}}(s) = \sum_{t=0}^{\infty} [\gamma^t \Pr(s_t = s|\boldsymbol{z})]$ *denotes the discounted visitation frequency of state* s *when conditioned on a particular input sequence* $\boldsymbol{z}$ *and policy* π_θ*, and* $Q(s_t, a_t|\boldsymbol{z}) = \mathbb{E}\left[\sum_{l=0}^{\infty} \gamma^l r(s_{t+l}, a_{t+l}) \,\middle|\, s_t, a_t, \boldsymbol{z}\right]$ *is the state-action value function under* $\boldsymbol{z}$ *and* π_θ*.*

Proof. See Appendix D.

We now show that using an input-dependent baseline to estimate the policy gradient does not introduce bias. This result is due to the conditional independence of the input process and the action a_t given the state s_t.

Theorem 1. *An input-dependent baseline* $b(s, \boldsymbol{z})$ *does not bias the Policy Gradient, i.e.,* $\mathbb{E}_{\boldsymbol{z},\rho_{\pi,\boldsymbol{z}},\pi_\theta} [\nabla_\theta \log \pi_\theta(a|s) b(s, \boldsymbol{z})] = 0$.

Proof. See Appendix E.

Input-dependent baselines are also bias-free for TRPO, as we show in Appendix G. Next, we derive the optimal input-dependent baseline in terms of variance reduction. As the gradient estimates are vectors, we use the trace of the covariance matrix to compute the variance (Greensmith et al., 2004).

Theorem 2. *The input-dependent baseline that minimizes variance in Policy Gradient is given by*

$$b^*(s, \boldsymbol{z}) = \frac{\mathbb{E}_{a\sim\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a|s)^T \nabla_\theta \log \pi_\theta(a|s) Q(s, a|\boldsymbol{z})\right]}{\mathbb{E}_{a\sim\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a|s)^T \nabla_\theta \log \pi_\theta(a|s)\right]}. \tag{5}$$

Proof. See Appendix F.

Operationally, for state s_t at each step t, the input-dependent baseline can take the form $b(s_t, z_{t:\infty})$, because $(s_t, z_{t:\infty})$ is a sufficient statistic of $(s_t, \boldsymbol{z})$ for $(s_{t:\infty}, a_{t:\infty}, z_{t:\infty})$. In practice, we use a simpler baseline $b(s_t, z_{t:\infty}) = \mathbb{E}_{a_t\sim\pi_\theta}[Q(s_t, a_t|z_{t:\infty})]$, which is the value function conditioned on the future input values $z_{t:\infty}$. We discuss how to estimate input-dependent baselines efficiently in §5.

Remark. Input-dependent baselines are generally applicable to reduce variance in policy gradient methods in input-driven environments. We apply input-dependent baselines to A2C (§6.2), TRPO (§6.1) and PPO (Appendix K). Also, our technique is complementary and orthogonal to adversarial RL (e.g., RARL (Pinto et al., 2017)) for environments with external disturbances. Those methods improve policy robustness by co-training an "adversary" to generate a worst-case disturbance process, whereas input-dependent baselines improves policy optimization *itself* in the presence of input processes like disturbances. In fact, input-dependent baselines can be used to improve the policy optimization step in adversarial RL methods. In Appendix L, we empirically show that if an adversary generates high-variance noise, RARL with standard state-based baseline is not adequate to train good controllers, and the input-dependent baseline helps improve the policy performance.

5 Learning Input-Dependent Baselines Efficiently

Input-dependent baselines are functions of the sequence of input values. A natural approach to train such baselines is to use models that operate on sequences (e.g., LSTMs (Gers et al., 1999)). However, learning a sequential mapping in a high-dimensional space can be expensive (Bahdanau et al., 2014). We considered an LSTM approach but ruled it out when initial experiments showed that it requires orders of magnitude more data to train than conventional baselines for our environments.

Fortunately, we can learn the baseline much more efficiently in applications where we can repeat the same input sequence multiple times during training. Input-repeatability is feasible in many

applications. For example, it is straightforward when using simulators for training. It is also applicable to training a real system using previously-collected input traces. For example, consider training a robot in the presence of exogenous forces. We could collect a large set of time-series traces of these forces, and apply them repeatedly to a physical robot for training. We now present two approaches that exploit input-repeatability to learn input-dependent baselines efficiently.

Multi-value-network approach. A straightforward way to learn $b(s_t, z_{t:\infty})$ for different input instantiations $\boldsymbol{z}$ is to train one value network to each particular instantiation of the input process. Specifically, in the training process, we first generate N input sequences $\{\boldsymbol{z}_1, \boldsymbol{z}_2, \cdots, \boldsymbol{z}_N\}$ and restrict training *only* to those N sequences. To learn a separate baseline function for each input sequence, we use N value networks with independent parameters $\theta_{V_1}, \theta_{V_2}, \cdots, \theta_{V_N}$, and single policy network with parameter θ. During training, we randomly sample an input sequence $\boldsymbol{z}_i$, execute a rollout based on $\boldsymbol{z}_i$ with the current policy π_θ, and use the (state, action, reward) data to train the value network parameter θ_{V_i} and the policy network parameter θ (details in Appendix H).

Meta-learning approach. The multi-value-network approach does not scale if the task requires training over a large number of input instantiations to generalize. Ideally, we would like an approach that enables shared learning across different input sequences. We present a different method based on *meta learning* to maximize the use of information across input sequences. The idea is to use *all* (potentially infinitely many) inputs sequences to learn a "meta value network" model. For each specific input sequence, we first customize the meta value network for that input sequence, using a few example rollouts with that input sequence. We then compute the actual baseline values for training the policy network parameters, using the customized value network for the specific input sequence. Our implementation uses Model-Agnostic Meta Learning (MAML) (Finn et al., 2017).

Algorithm 1 Training a meta input-dependent baseline for policy-based methods.

Require: α, β: meta value network step size hyperparameters
1: Initialize policy network parameters θ and meta value network parameters θ_V
2: **while** not done **do**
3: Generate a new input sequence $\boldsymbol{z}$
4: Sample k rollouts $\mathcal{T}_1, \mathcal{T}_2, ..., \mathcal{T}_k$ using policy π_θ and input sequence $\boldsymbol{z}$
5: Adapt θ_V with the first $k/2$ rollouts: $\theta_V^1 = \theta_V - \alpha \nabla_{\theta_V} \mathcal{L}_{\mathcal{T}_{1:k/2}} [V_{\theta_V}]$
6: Estimate baseline value $V_{\theta_V^1}(s_t)$ for $s_t \sim \mathcal{T}_{k/2:k}$ using adapted θ_V^1
7: Adapt θ_V with the second $k/2$ rollouts: $\theta_V^2 = \theta_V - \alpha \nabla_{\theta_V} \mathcal{L}_{\mathcal{T}_{k/2:k}} [V_{\theta_V}]$
8: Estimate baseline value $V_{\theta_V^2}(s_t)$ for $s_t \sim \mathcal{T}_{1:k/2}$ using adapted θ_V^2
9: Update policy with Equation (1) or (2) using the values from line (6) and (8) as baseline
10: Update meta value network: $\theta_V \leftarrow \theta_V - \beta \nabla_{\theta_V} \mathcal{L}_{k/2:k} \left[V_{\theta_V^1}\right] - \beta \nabla_{\theta_V} \mathcal{L}_{1:k/2} \left[V_{\theta_V^2}\right]$
11: **end while**

The pseudocode in Algorithm 1 depicts the training algorithm. We follow the notation of MAML, denoting the loss in the value function $V_{\theta_V}(\cdot)$ on a rollout $\mathcal{T}$ as $\mathcal{L}_{\mathcal{T}}[V_{\theta_V}] = \sum_{s_t, r_t \sim \mathcal{T}} \| V_{\theta_V}(s_t) - \sum_{t'=t}^{T} \gamma^{t'-t} r_t \|^2$. We perform rollouts k times with the same input sequence $\boldsymbol{z}$ (lines 3 and 4); we use the first $k/2$ rollouts to customize the meta value network for this instantiation of $\boldsymbol{z}$ (line 5), and then apply the customized value network on the states of the other $k/2$ rollouts to compute the baseline for those rollouts (line 6); similarly, we swap the two groups of rollouts and repeat the same process (lines 7 and 8). We do not use the same rollouts to adapt the meta value network and compute the baseline to avoid introducing extra bias to the baseline. Finally, we use the baseline values computed for each rollout to update the policy network parameters (line 9), and we apply the MAML (Finn et al., 2017) gradient step to update the meta value network model (line 10).

6 Experiments

Our experiments demonstrate that input-dependent baselines provide consistent performance gains across multiple continuous-action MuJoCo simulated robotic locomotions and discrete-action environments in queuing systems and network control. We conduct experiments for both policy gradient methods and policy optimization methods (see Appendix J for details). The videos for our experiments are available at https://sites.google.com/view/input-dependent-baseline/.

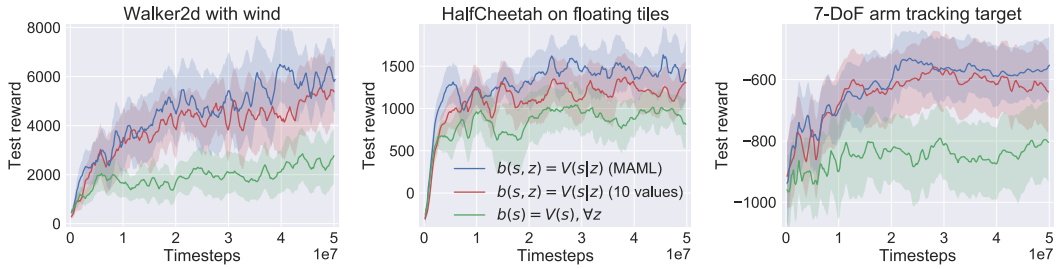

Figure 4: In continuous-action MuJoCo environments, TRPO (Schulman et al., 2015a) with input-dependent baselines achieve 25%–3× better testing reward than that with a standard state-dependent baseline. Learning curves are on 100 testing episodes with unseen input sequences; shaded area spans one standard deviation.

6.1 Simulated Robotic Locomotion

We use the MuJoCo physics engine (Todorov et al., 2012) in OpenAI Gym (Brockman et al., 2016) to evaluate input-dependent baselines for robotic control tasks with external disturbance. We extend the standard walker-2d, half-cheetah and 7-DoF robotic arm environments, adding a different external input to each (Figure 1).

Walker2d with random wind (Figure 1c). A 2D walker is trained with varying wind, which randomly drags the walker backward or forward with different force at each step. The wind vector changes randomly, i.e., the wind forms a random input process. We add a force sensor to the state to enable the agent to quickly adapt. The goal is for the walker to walk forward while keeping balance.

HalfCheetah on floating tiles with random buoyancy (Figure 1d). A half-cheetah runs over a series of tiles floating on water (Clavera et al., 2018). Each tile has different damping and friction properties, which moves the half-cheetah up and down and changes its dynamics. This random buoyancy is the external input process; the cheetah needs to learn running forward over varying tiles.

7-DoF arm tracking moving target (Figure 1e). We train a simulated robot arm to track a randomly moving target (a red ball). The robotic arm has seven degrees of freedom and the target is doing a random walk, which forms the external input process. The reward is the negative squared distance between the robot hand (blue square) and the target.

Results. We build 10-value networks and a meta-baseline using MAML, both on top of the OpenAI's TRPO implementation (Dhariwal et al., 2017). Figure 4 shows the performance comparison among different baselines with 100 *unseen* testing input sequences at each training checkpoint. These learning curves show that TRPO with a state-dependent baseline performs worst in all environments. With the input-dependent baseline, by contrast, performance in unseen testing environment improves by up to 3×. The agent is able to learn a policy robust against disturbances. For example, it learns to lean into headwind and quickly place its leg forward to counter the headwind; it learns to apply different force on tiles with different buoyancy to avoid falling over; and it learns to co-adjust multiple joints to keep track of the moving object. The meta-baseline eventually outperforms 10-value networks as it effectively learns from a large number of input processes and hence generalizes better.

The input-dependent baseline technique applies generally on top of policy optimization methods. In Appendix K, we show a similar comparison with PPO (Schulman et al., 2017). Also, in Appendix L we show that adversarial RL (e.g., RARL (Pinto et al., 2017)) alone is not adequate to solve the high variance problem, and the input-dependent baseline helps improve the policy performance (Figure 7).

6.2 Discrete-Action Environments

Our discrete-action environments arise from widely-studied problems in computer systems research: load balancing and bitrate adaptation.[1] As these problems often lack closed-form optimal solutions (Grandl et al., 2016; Yin et al., 2015), hand-tuned heuristics abound. Recent work suggests that model-free reinforcement learning can achieve better performance than such human-engineered

[1] We considered Atari games often used as benchmark discrete-action RL environments (Mnih et al., 2015). However, Atari games are not applicable to our work, as they lack an exogenous input process: a random seed perturbs the games' initial state, but it does not control the pattern of environmental changes (e.g., in "Seaquest", the ships always come in a fixed pattern).

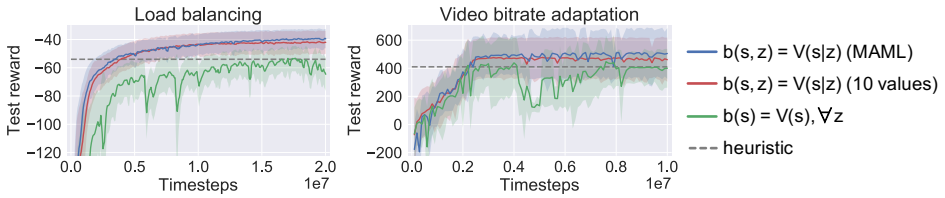

Figure 5: In environments with discrete action spaces, A2C (Mnih et al., 2016) with input-dependent baselines outperform the best heuristic and achieve 25–33% better testing reward than vanilla A2C (Mnih et al., 2016). Learning curves are on 100 test episodes with unseen input sequences; shaded area spans one standard deviation.

heuristics (Mao et al., 2016; Evans & Gao, 2016; Mao et al., 2017; Mirhoseini et al., 2017). We consider a load balancing environment (similar to the example in §3) and a bitrate adaptation environment in video streaming (Yin et al., 2015). The detailed setup of these environments is in Appendix I.

Results. We extend OpenAI's A2C implementation (Dhariwal et al., 2017) for our baselines. The learning curves in Figure 5 illustrate that directly applying A2C with a standard value network as the baseline results in unstable test reward and underperforms the traditional heuristic in both environments. Our input-dependent baselines reduce the variance and improve test reward by 25–33%. The meta-baseline performs the best in all environments.

7 Related Work

Policy gradient methods compute unbiased gradient estimates, but can experience a large variance (Sutton & Barto, 1998; Weaver & Tao, 2001). Reducing variance for policy-based methods using a baseline has been shown to be effective (Williams, 1992; Sutton & Barto, 1998; Weaver & Tao, 2001; Greensmith et al., 2004; Mnih et al., 2016). Much of this work focuses on variance reduction in a general MDP setting, rather than variance reduction for MDPs with specific stochastic structures. Wu et al. (2018)'s techniques for MDPs with multi-variate independent actions are closest to our work. Their state-action-dependent baseline improves training efficiency and model performance on high-dimensional control tasks by explicitly factoring out, for each action, the effect due to other actions. By contrast, our work exploits the structure of state transitions instead of stochastic policy.

Recent work has also investigated the bias-variance tradeoff in policy gradient methods. Schulman et al. (2015b) replace the Monte Carlo return with a λ-weighted return estimation (similar to TD(λ) with value function bootstrap (Tesauro, 1995)), improving performance in high-dimensional control tasks. Other recent approaches use more general control variates to construct variants of policy gradient algorithms. Tucker et al. (2018) compare the recent work, both analytically on a linear-quadratic-Gaussian task and empirically on complex robotic control tasks. Analysis of control variates for policy gradient methods is a well-studied topic, and extending such analyses (e.g., Greensmith et al. (2004)) to the input-driven MDP setting could be interesting future work.

In other contexts, prior work has proposed new RL training methodologies for environments with disturbances. Clavera et al. (2018) adapts the policy to different pattern of disturbance by training the RL agent using meta-learning. RARL (Pinto et al., 2017) improves policy robustness by co-training an adversary to generate a worst-case noise process. Our work is orthogonal and complementary to these work, as we seek to improve policy optimization itself in the presence of inputs like disturbances.

8 Conclusion

We introduced input-driven Markov Decision Processes in which stochastic input processes influence state dynamics and rewards. In this setting, we demonstrated that an input-dependent baseline can significantly reduce variance for policy gradient methods, improving training stability and the quality of learned policies. Our work provides an important ingredient for using RL successfully in a variety of domains, including queuing networks and computer systems, where an input workload is a fundamental aspect of the system, as well as domains where the input process is more implicit, like robotics control with disturbances or random obstacles.

We showed that meta-learning provides an efficient way to learn input-dependent baselines for applications where input sequences can be repeated during training. Investigating efficient architectures for input-dependent baselines for cases where the input process cannot be controlled in training is an interesting direction for future work.

REFERENCES

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. https://gym.openai.com/docs/, 2016.

Trishul Chilimbi, Yutaka Suzue, Johnson Apacible, and Karthik Kalyanaraman. Project adam: Building an efficient and scalable deep learning training system. In *OSDI*, pp. 571–582, Broomfield, CO, October 2014. USENIX Association.

Ignasi Clavera, Anusha Nagabandi, Ronald S Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt: Meta-learning for model-based control. *arXiv preprint arXiv:1803.11347*, 2018.

DJ Daley. Certain optimality properties of the first-come first-served discipline for g/g/s queues. *Stochastic Processes and their Applications*, 25:301–308, 1987.

DASH Industry Form. Reference Client 2.4.0. http://mediapm.edgesuite.net/dash/public/nightly/samples/dash-if-reference-player/index.html, 2016.

Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, and Yuhuai Wu. Openai baselines. https://github.com/openai/baselines, 2017.

Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning*, pp. 1329–1338, 2016.

Richard Evans and Jim Gao. DeepMind AI Reduces Google Data Centre Cooling Bill by 40%. https://deepmind.com/blog/deepmind-ai-reduces-google-data-centre-cooling-bill-40/, 2016.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pp. 1126–1135, 2017.

Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. 1999.

Robert Grandl, Srikanth Kandula, Sriram Rao, Aditya Akella, and Janardhan Kulkarni. Graphene: Packing and dependency-aware scheduling for data-parallel clusters. In *Proceedings of OSDI*, pp. 81–97. USENIX Association, 2016.

Evan Greensmith, Peter L Bartlett, and Jonathan Baxter. Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research*, 5(Nov):1471–1530, 2004.

Shixiang Gu, Tim Lillicrap, Richard E Turner, Zoubin Ghahramani, Bernhard Schölkopf, and Sergey Levine. Interpolated policy gradient: Merging on-policy and off-policy gradient estimation for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 3849–3858, 2017.

Mor Harchol-Balter and Rein Vesilo. To balance or unbalance load in size-interval task allocation. *Probability in the Engineering and Informational Sciences*, 24(2):219–244, April 2010.

Nicolas Heess, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, Ali Eslami, Martin Riedmiller, et al. Emergence of locomotion behaviours in rich environments. *arXiv preprint arXiv:1707.02286*, 2017.

Sham M Kakade. A natural policy gradient. In *Advances in neural information processing systems*, pp. 1531–1538, 2002.

Frank P Kelly. *Reversibility and stochastic networks*. Cambridge University Press, 2011.

Leonard Kleinrock. *Queueing systems, volume 2: Computer applications*, volume 66. wiley New York, 1976.

Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(1):1334–1373, January 2016.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Hongzi Mao, Mohammad Alizadeh, Ishai Menache, and Srikanth Kandula. Resource management with deep reinforcement learning. In *Proceedings of the 15th ACM Workshop on Hot Topics in Networks (HotNets)*, Atlanta, GA, November 2016.

Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. Neural adaptive video streaming with pensieve. In *Proceedings of the ACM SIGCOMM 2017 Conference*. ACM, 2017.

A Stephen McGough, Noura Al Moubayed, and Matthew Forshaw. Using machine learning in trace-driven energy-aware simulations of high-throughput computing systems. In *Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering Companion*, pp. 55–60. ACM, 2017.

Azalia Mirhoseini, Hieu Pham, Quoc V Le, Benoit Steiner, Rasmus Larsen, Yuefeng Zhou, Naveen Kumar, Mohammad Norouzi, Samy Bengio, and Jeff Dean. Device placement optimization with reinforcement learning. In *Proceedings of The 33rd International Conference on Machine Learning*, 2017.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, Demis Hassabis Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Tim Harley, Timothy P. Lillicrap, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, pp. 1928–1937, 2016.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814, 2010.

Art B. Owen. *Monte Carlo theory, methods and examples*. 2013.

Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. In *International Conference on Machine Learning*, pp. 2817–2826, 2017.

Haakon Riiser, Paul Vigmostad, Carsten Griwodz, and Pål Halvorsen. Commute Path Bandwidth Traces from 3G Networks: Analysis and Applications. In *Proceedings of the 4th ACM Multimedia Systems Conference*, MMSys. ACM, 2013.

John Schulman, Sergey Levine, Philipp Moritz, Michael I Jordan, and Pieter Abbeel. Trust region policy optimization. *CoRR, abs/1502.05477*, 2015a.

John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015b.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

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

R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

Richard S. Sutton, David A. McAllester, Satinder P. Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*, volume 99, pp. 1057–1063, 1999.

Gerald Tesauro. Temporal difference learning and td-gammon. *Communications of the ACM*, 38(3): 58–68, 1995.

Philip Thomas. Bias in natural actor-critic algorithms. In *International Conference on Machine Learning*, pp. 441–448, 2014.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 5026–5033. IEEE, 2012.

George Tucker, Surya Bhupatiraju, Shixiang Gu, Richard E Turner, Zoubin Ghahramani, and Sergey Levine. The mirage of action-dependent baselines in reinforcement learning. *arXiv preprint arXiv:1802.10031*, 2018.

Ricardo Vilalta and Youssef Drissi. A perspective view and survey of meta-learning. *Artificial Intelligence Review*, 18(2):77–95, 2002.

Lex Weaver and Nigel Tao. The optimal reward baseline for gradient-based reinforcement learning. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, pp. 538–545. Morgan Kaufmann Publishers Inc., 2001.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

Keith Winstein and Hari Balakrishnan. Tcp ex machina: Computer-generated congestion control. In *ACM SIGCOMM Computer Communication Review*, volume 43, pp. 123–134. ACM, 2013.

Cathy Wu, Aravind Rajeswaran, Yan Duan, Vikash Kumar, Alexandre M Bayen, Sham Kakade, Igor Mordatch, and Pieter Abbeel. Variance reduction for policy gradient with action-dependent factorized baselines. In *International Conference on Learning Representations*, 2018.

Yuxin Wu and Yuandong Tian. Training agent for first-person shooter game with actor-critic curriculum learning. In *Submitted to International Conference on Learning Representations*, 2017.

Xiaoqi Yin, Abhishek Jindal, Vyas Sekar, and Bruno Sinopoli. A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, SIGCOMM. ACM, 2015.

A ILLUSTRATION OF VARIANCE REDUCTION IN 1D GRID WORLD

Consider a walker in a 1D grid world, where the state $s_t \in \mathbb{Z}$ at time t denotes the position of the walker, and action $a_t \in \{-1, +1\}$ denotes the intent to either move forward or backward. Additionally let $z_t \in \{-1, +1\}$ be a uniform i.i.d. "exogenous input" that perturbs the position of the walker. For an action a_t and input z_t, the state of the walker in the next step is given by $s_{t+1} = s_t + a_t + z_t$. The objective of the game is to move the walker forward; hence, the reward is $r_t = a_t + z_t$ at each time step. $\gamma \in [0, 1]$ is a discount factor.

While the optimal policy for this game is clear ($a_t = +1$ for all t), consider learning such a policy using policy gradient. For simplicity, let the policy be parametrized as $\pi_\theta(a_t = +1|s_t) = e^\theta/(1+e^\theta)$, with θ initialized to 0 at the start of training. In the following we evaluate the variance of the policy gradient estimate at the start of training under (i) the standard value function baseline, and (ii) a baseline that is the expected cumulative reward conditioned on all future z_t inputs.

Variance under standard baseline. The value function in this case is identically 0 at all states. This is because $\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t] = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t (a_t + z_t)] = 0$ since both actions a_t and inputs z_t are i.i.d. with mean 0. Also note that $\nabla_\theta \log \pi_\theta(a_t = +1) = 1/2$ and $\nabla_\theta \log \pi_\theta(a_t = -1) = -1/2$; hence $\nabla_\theta \log \pi_\theta(a_t) = a_t/2$. Therefore the variance of the policy gradient estimate can be written as

$$V_1 = \mathrm{Var}\left[\sum_{t=0}^{\infty} \frac{a_t}{2} \sum_{t'=t}^{\infty} \gamma^{t'} r_{t'}\right] = \mathrm{Var}\left[\sum_{t=0}^{\infty} \frac{a_t}{2} \sum_{t'=t}^{\infty} \gamma^{t'} (a_{t'} + z_{t'})\right]. \tag{6}$$

Variance under input-dependent baseline. Now, consider an alternative "input-dependent" baseline $V(s_t|\boldsymbol{z})$ defined as $\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t|\boldsymbol{z}]$. Intuitively this baseline captures the average reward incurred when experiencing a particular fixed $\boldsymbol{z}$ sequence. We refer the reader to §4 for a formal discussion and analysis of input-dependent baselines. Evaluating the baseline we get $V(s_t|\boldsymbol{z}) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t|\boldsymbol{z}] = \sum_{t=0}^{\infty} \gamma^t z_t$. Therefore the variance of the policy gradient estimate in this case is

$$V_2 = \mathrm{Var}\left[\sum_{t=0}^{\infty} \frac{a_t}{2} \left(\sum_{t'=t}^{\infty} \gamma^{t'} r_{t'} - \sum_{t'=t}^{\infty} \gamma^{t'} z_{t'}\right)\right] = \mathrm{Var}\left[\sum_{t=0}^{\infty} \frac{a_t}{2} \left(\sum_{t'=t}^{\infty} \gamma^{t'} a_{t'}\right)\right]. \tag{7}$$

Reduction in variance. To analyze the variance reduction between the two cases (Equations (6) and (7)), we note that

$$V_1 = V_2 + \mathrm{Var}\left[\sum_{t=0}^{\infty} \frac{a_t}{2} \left(\sum_{t'=t}^{\infty} \gamma^{t'} z_{t'}\right)\right] + 2\mathrm{Cov}\left(\sum_{t=0}^{\infty} \frac{a_t}{2} \left(\sum_{t'=t}^{\infty} \gamma^{t'} a_{t'}\right), \sum_{t=0}^{\infty} \frac{a_t}{2} \left(\sum_{t'=t}^{\infty} \gamma^{t'} z_{t'}\right)\right) \tag{8}$$

$$= V_2 + \mathrm{Var}\left[\sum_{t=0}^{\infty} \frac{a_t}{2} \left(\sum_{t'=t}^{\infty} \gamma^{t'} z_{t'}\right)\right]. \tag{9}$$

This follows because

$$\mathbb{E}\left[\sum_{t=0}^{\infty} \frac{a_t}{2} \left(\sum_{t'=t}^{\infty} \gamma^{t'} z_{t'}\right)\right] = \sum_{t=0}^{\infty} \sum_{t'=t}^{\infty} \frac{\gamma^{t'}}{2} \mathbb{E}[a_t z_{t'}] = 0, \quad \text{and}$$

$$\mathbb{E}\left[\left(\sum_{t=0}^{\infty} \frac{a_t}{2} \left(\sum_{t'=t}^{\infty} \gamma^{t'} a_{t'}\right)\right) \left(\sum_{t=0}^{\infty} \frac{a_t}{2} \left(\sum_{t'=t}^{\infty} \gamma^{t'} z_{t'}\right)\right)\right] =$$

$$\sum_{t_1=0}^{\infty} \sum_{t_1'=t_1}^{\infty} \sum_{t_2=0}^{\infty} \sum_{t_2'=t_2}^{\infty} \mathbb{E}\left[\frac{a_{t_1} a_{t_1'} a_{t_2} z_{t_2'}}{4} \gamma^{t_1'+t_2'}\right] = 0.$$

Therefore the covariance term in Equation (8) is 0. Hence the variance reduction from Equation (9) can be written as

$$V_1 - V_2 = \mathrm{Var}\left[\sum_{t=0}^{\infty} \frac{a_t}{2} \left(\sum_{t'=t}^{\infty} \gamma^{t'} z_{t'}\right)\right] = \sum_{t_1=0}^{\infty} \sum_{t_1'=t_1}^{\infty} \sum_{t_2=0}^{\infty} \sum_{t_2'=t_2}^{\infty} \mathbb{E}\left[\frac{a_{t_1} a_{t_2} z_{t_1'} z_{t_2'}}{4} \gamma^{t_1'+t_2'}\right]$$

$$= \sum_{t_1=0}^{\infty} \sum_{t_1'=t_1}^{\infty} \mathbb{E}\left[\frac{a_{t_1}^2 z_{t_1'}^2}{4} \gamma^{2t_1'}\right] = \frac{\mathrm{Var}(a_0)\mathrm{Var}(z_0)}{4(1-\gamma^2)^2}.$$

Thus the input-dependent baseline reduces variance of the policy gradient estimate by an amount proportional to the variance of the external input. While in this toy example we have chosen z_t to be binary-valued, more generally the variance of z_t could be arbitrarily large and be a dominating factor of the overall variance in the policy gradient estimation.

B Proof of Proposition 1

Proof. Let $(\mathcal{S}, \mathcal{A}, \mathcal{Z}, \mathcal{P}_s, \mathcal{P}_z, \rho_0^s, \rho_0^z, r, \gamma)$ be an input-driven MDP, defined in Definition 1.

For case 1 (Figure 3a), let $\tilde{s}_t := (s_t, z_t)$ and $\tilde{a}_t := a_t$. We seek to show that the process formed by $\{\tilde{s}_t, \tilde{a}_t\}_{t=0}^{\infty}$ forms a standard MDP. Clearly $\tilde{s}_t \in \mathcal{S} \times \mathcal{Z}$, and $\tilde{\rho}_0(\tilde{s}_0) = \Pr(s_0, z_0) = \rho_0^s(s_0)\rho_0^z(z_0)$ Notice that the definition of an input-driven MDP presumes the Markov property

$$\Pr(s_{t+1}, z_{t+1}|s_{0:t}, z_{0:t}, a_{0:t}) = \Pr(s_{t+1}, z_{t+1}|s_t, z_t, a_t) \tag{10}$$

$$= \mathcal{P}_s(s_{t+1}|s_t, z_t, a_t)\mathcal{P}_z(z_{z+1}|z_t) \quad \forall t \geq 0. \tag{11}$$

As such, the process $\{\tilde{s}_t, \tilde{a}_t\}_{t=0}^{\infty}$ is also Markov and satisfies

$$\Pr(\tilde{s}_{t+1}|\tilde{s}_{0:t}, \tilde{a}_{0:t}) = \Pr(\tilde{s}_{t+1}|\tilde{s}_t, \tilde{a}_t) \quad \forall t \geq 0. \tag{12}$$

Finally to specify the transition kernel for $\{\tilde{s}_t, \tilde{a}_t\}_{t=0}^{\infty}$ we have from Equation (11)

$$\tilde{\mathcal{P}}(\tilde{s}_{t+1}|\tilde{s}_t, \tilde{a}_t) = \Pr(s_{t+1}, z_{t+1}|s_t, z_t, a_t) = \mathcal{P}_s(s_{t+1}|s_t, z_t, a_t)\mathcal{P}_z(z_{z+1}|z_t) \tag{13}$$

Therefore, for case 1, $(\tilde{\mathcal{S}}, \mathcal{A}, \tilde{\mathcal{P}}, \tilde{\rho}_0, r, \gamma)$ defines a standard MDP, where $\tilde{\mathcal{S}} = \mathcal{S} \times \mathcal{Z}$, $\tilde{\mathcal{P}}(s_{t+1}, z_{t+1}|s_t, z_t, a_t) = \mathcal{P}_s(s_{t+1}|s_t, a_t, z_t)\mathcal{P}_z(z_{t+1}|z_t)$, and $\tilde{\rho}_0(s_0, z_0) = \rho_0^s(s_0)\rho_0^z(z_0)$.

Similarly, for case 2 (Figure 3b), consider the process $\{s_t, a_t\}$. Then clearly $s_t \in \mathcal{S}$, $a_t \in \mathcal{A}$ and $s_0 \sim \rho_0^s$.

Now,

$$\Pr(s_{t+1}|s_{0:t}, a_{0:t}) = \sum_{z_{0:t}} \Pr(z_{0:t}) \Pr(s_{t+1}|s_{0:t}, a_{0:t}, z_{0:t}) \tag{14}$$

$$= \sum_{z_{0:t}} \Pr(z_{0:t})\mathcal{P}_s(s_{t+1}|s_t, a_t, z_t) \quad \text{(from Equation 11)} \tag{15}$$

$$= \sum_{z_{0:t}} \mathcal{P}_s(s_{t+1}|s_t, a_t, z_t) \Pr(z_t) \Pr(z_{t-1}) \ldots \quad \text{(because } z_t \text{ is i.i.d.)} \tag{16}$$

$$= \sum_{z_t} \mathcal{P}_s(s_{t+1}|s_t, a_t, z_t) \Pr(z_t) \sum_{z_{t-1}} \Pr(z_{t-1}) \sum_{z_{t-2}} \Pr(z_{t-2}) \ldots \tag{17}$$

$$= \sum_{z_t} \mathcal{P}_s(s_{t+1}|s_t, a_t, z_t) \Pr(z_t) \tag{18}$$

$$= \Pr(s_{t+1}|s_t, a_t). \tag{19}$$

Therefore $\{s_t, a_t\}_{t=0}^{\infty}$ forms the MDP $(\mathcal{S}, \mathcal{A}, \bar{\mathcal{P}}, \rho_0^s, r, \gamma)$ with a transition kernel $\bar{\mathcal{P}}$ as given by Equation (18).

□

C Proof of Lemma 1

Proof. From the definition of an input-driven MDP (Definition 1), we have

$$\begin{aligned}
\Pr(\boldsymbol{z}, s_t, a_t) &= \Pr(z_{0:t}, s_t, a_t) \Pr(z_{t+1:\infty}|z_{0:t}, s_t, a_t) \\
&= \Pr(z_{0:t}) \Pr(s_t|z_{0:t}) \Pr(a_t|s_t, z_{0:t}) \Pr(z_{t+1:\infty}|z_t) \\
&= \Pr(z_{0:t}) \Pr(s_t|z_{0:t})\pi_\theta(a_t|s_t) \Pr(z_{t+1:\infty}|z_{0:t}) \\
&= \Pr(\boldsymbol{z}) \Pr(s_t|z_{0:t})\pi_\theta(a_t|s_t) \\
&= \Pr(\boldsymbol{z}) \Pr(s_t|\boldsymbol{z})\pi_\theta(a_t|s_t).
\end{aligned}$$

□

D PROOF OF LEMMA 2

Proof. Expanding the Policy Gradient Theorem in Equation (1), we have

$$
\begin{aligned}
\nabla_\theta \eta(\pi_\theta) =& \mathbb{E}\left[\sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(a_t|s_t) \sum_{t' \geq t} \gamma^{t'} r(s_{t'}, a_{t'})\right] \\
=& \sum_{t=0}^{\infty} \mathbb{E}\left[\nabla_\theta \log \pi_\theta(a_t|s_t) \sum_{t' \geq t} \gamma^{t'} r(s_{t'}, a_{t'})\right] \\
=& \sum_{t=0}^{\infty} \left[\sum_{\boldsymbol{z},s,a} \Pr(\boldsymbol{z}) \Pr(s_t = s|\boldsymbol{z}) \pi_\theta(a_t = a|s) \nabla_\theta \log \pi_\theta(a|s) \mathbb{E}\left[\sum_{t'=t} \gamma^{t'} r(s_{t'}, a_{t'}) | \boldsymbol{z}, s_t = s, a_t = a\right]\right] \qquad (20) \\
=& \sum_{t=0}^{\infty} \left[\sum_{\boldsymbol{z},s,a} \Pr(\boldsymbol{z}) \Pr(s_t = s|\boldsymbol{z}) \pi_\theta(a|s) \nabla_\theta \log \pi_\theta(a|s) \gamma^t Q(s, a|\boldsymbol{z})\right] \qquad (21) \\
=& \sum_{\boldsymbol{z},s,a} \left[\Pr(\boldsymbol{z}) \pi_\theta(a|s) \nabla_\theta \log \pi_\theta(a|s) Q(s, a|\boldsymbol{z}) \left[\sum_{t=0}^{\infty} \gamma^t \Pr(s_t = s|\boldsymbol{z})\right]\right] \\
=& \sum_{\boldsymbol{z},s,a} \left[\Pr(\boldsymbol{z}) \rho_{\pi,\boldsymbol{z}}(s) \pi_\theta(a|s) \nabla_\theta \log \pi_\theta(a|s) Q(s, a|\boldsymbol{z})\right] \qquad (22) \\
=& \mathbb{E}_{\boldsymbol{z}, \rho_{\pi,\boldsymbol{z}}, \pi_\theta}\left[\nabla_\theta \log \pi_\theta(a|s) Q(s, a|\boldsymbol{z})\right],
\end{aligned}
$$

where in Equation (20) we have used the fact that the policy is conditionally independent of the input-sequence given the present state (Lemma 1), while Equations (21) and (22) follow from the definitions of the Q-function and state-visitation frequency $\rho_{\pi,\boldsymbol{z}}$ respectively. □

E PROOF OF THEOREM 1

Proof. For any state $s \in \mathcal{S}$, we first note that $\sum_{a \in \mathcal{A}} \nabla_\theta \pi_\theta(a|s) = \nabla_\theta \sum_{a \in \mathcal{A}} \pi_\theta(a|s) = 0$. As such, we have

$$
\begin{aligned}
\mathbb{E}_{\boldsymbol{z}, \rho_{\pi,\boldsymbol{z}}, \pi_\theta}\left[\nabla_\theta \log \pi_\theta(a|s) b(s, \boldsymbol{z})\right] =& \mathbb{E}_{\boldsymbol{z}}\left[\mathbb{E}_{\rho_{\pi,\boldsymbol{z}}, \pi_\theta}\left[\nabla_\theta \log \pi_\theta(a|s) b(s, \boldsymbol{z}) \mid \boldsymbol{z}\right]\right] \\
=& \mathbb{E}_{\boldsymbol{z}}\left[\sum_s \sum_a \rho_{\pi,\boldsymbol{z}}(s) \pi_\theta(a|s) \nabla_\theta \log \pi_\theta(a|s) b(s, \boldsymbol{z})\right] \\
=& \mathbb{E}_{\boldsymbol{z}}\left[\sum_s \rho_{\pi,\boldsymbol{z}}(s) b(s, \boldsymbol{z}) \sum_a \nabla_\theta \pi_\theta(a|s)\right] = 0. \qquad (23)
\end{aligned}
$$

□

Notice that the proof above (Equation (23)) again relies on the Markov property of $\boldsymbol{z} - s_t - a_t$.

F PROOF OF THEOREM 2

Proof. Let $G(s,a)$ denote $\nabla_\theta \log \pi_\theta(a|s)^T \nabla_\theta \log \pi_\theta(a|s)$. For any input-aware baseline $b(s,\boldsymbol{z})$, the variance of the Policy Gradient estimate is given by

$$
\begin{aligned}
&\mathbb{E}_{\substack{\boldsymbol{z}\sim P_{\boldsymbol{z}} \\ s\sim\rho_{\pi,\boldsymbol{z}} \\ a\sim\pi_\theta}}\left[\left\|\nabla_\theta \log \pi_\theta(a|s)\left[Q(s,a|\boldsymbol{z}) - b(s,\boldsymbol{z})\right] - \mathbb{E}_{\boldsymbol{z},\rho_{\pi,\boldsymbol{z}},\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a|s)\left[Q(s,a|\boldsymbol{z}) - b(s,\boldsymbol{z})\right]\right]\right\|_2^2\right] \\
&= \mathbb{E}_{\boldsymbol{z},\rho_{\pi,\boldsymbol{z}},\pi_\theta}\left[G(s,a)\left[Q(s,a|\boldsymbol{z}) - b(s,\boldsymbol{z})\right]^2\right] - \left\|\mathbb{E}_{\boldsymbol{z},\rho_{\pi,\boldsymbol{z}},\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a|s)\left[Q(s,a|\boldsymbol{z}) - b(s,\boldsymbol{z})\right]\right]\right\|_2^2 \\
&= \mathbb{E}_{\boldsymbol{z},\rho_{\pi,\boldsymbol{z}},\pi_\theta}\left[G(s,a)\left[Q(s,a|\boldsymbol{z}) - b(s,\boldsymbol{z})\right]^2\right] - \left\|\mathbb{E}_{\boldsymbol{z},\rho_{\pi,\boldsymbol{z}},\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a|s) Q(s,a|\boldsymbol{z})\right]\right\|_2^2 \quad \text{(due to Theorem 1)} \\
&= \mathbb{E}_{\boldsymbol{z},\rho_{\pi,\boldsymbol{z}},\pi_\theta}\left[G(s,a)Q(s,a|\boldsymbol{z})^2\right] - \left\|\mathbb{E}_{\boldsymbol{z},\rho_{\pi,\boldsymbol{z}},\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a|s) Q(s,a|\boldsymbol{z})\right]\right\|_2^2 \\
&\quad + \mathbb{E}_{\boldsymbol{z},\rho_{\pi,\boldsymbol{z}}}\left[\mathbb{E}_{a\sim\pi_\theta}\left[G(s,a) \mid \boldsymbol{z}, s\right] b(s,\boldsymbol{z})^2 - 2\mathbb{E}_{a\sim\pi_\theta}\left[G(s,a)Q(s,a|\boldsymbol{z}) \mid \boldsymbol{z}, s\right] b(s,\boldsymbol{z})\right].
\end{aligned}
$$

Notice that the baseline is only involved in the last term in a quadratic form, where the second order term is positive. To minimize the variance, we set baseline to the minimizer of the quadratic equation, i.e., $2\mathbb{E}_{a\sim\pi_\theta}\left[G(s,a) \mid \boldsymbol{z}, s\right] b(s,\boldsymbol{z}) - 2\mathbb{E}_{a\sim\pi_\theta}\left[G(s,a)Q(s,a|\boldsymbol{z}) \mid \boldsymbol{z}, s\right] = 0$ and hence the result follows. □

G INPUT-DEPENDENT BASELINE FOR TRPO

We show that the input-dependent baselines are bias-free for TRPO (Schulman et al., 2015a).

Theorem 3. *$argmax_\theta \mathbb{E}_{s\sim\rho_{old},a\sim\pi_{old}}\left[\frac{\pi_\theta(a|s)}{\pi_{old}(a|s)}\hat{Q}_{\pi_{old}}(s,a)\right] = argmax_\theta \mathbb{E}_{s\sim\rho_{old},a\sim\pi_{old}}\left[\frac{\pi_\theta(a|s)}{\pi_{old}(a|s)}\left(\hat{Q}_{\pi_{old}}(s,a) - b(s,\boldsymbol{z})\right)\right]$, which means the baseline does not change the optimization objective.*

Proof.

$$
\begin{aligned}
&\mathbb{E}_{s\sim\rho_{\text{old}},a\sim\pi_{\text{old}}}\left[\frac{\pi_\theta(a|s)}{\pi_{\text{old}}(a|s)} b(s,\boldsymbol{z})\right] \\
&= \sum_s \rho_{\text{old}}(s) \sum_a \pi_{\text{old}}(a|s)\left[\frac{\pi_\theta(a|s)}{\pi_{\text{old}}(a|s)} b(s,\boldsymbol{z})\right] \\
&= \sum_s \rho_{\text{old}}(s) \sum_a \pi_\theta(a|s) b(s,\boldsymbol{z}) \\
&= \sum_s \rho_{\text{old}}(s) b(s,\boldsymbol{z}),
\end{aligned}
$$

which is independent of θ. Therefore, $b(s,\boldsymbol{z})$ does not change the optimization objective. □

H PSEUDOCODE FOR TRAINING MULTI-VALUE BASELINES

In §5 we explained the idea of efficiently computing input-dependent baselines (§4.1) using multiple value networks on a fixed set of input sequences. Algorithm 2 depicts the details of this approach.

Algorithm 2 Training multi-value baselines for policy-based methods.

Require: pregenerated input seuqnces $\{\boldsymbol{z}_1, \boldsymbol{z}_2, \cdots, \boldsymbol{z}_N\}$, step sizes α, β

```
Initialize value network parameters θ_{V_1}, θ_{V_1}, ···, θ_{V_N} and policy parameters θ
while not done do
    Sample a input sequence z_i
    Sample k rollouts T_1, T_2, ..., T_k using policy π_θ and input sequence z_i
    Update policy with Equation (1) or (2) using baseline estimated with θ_{V_i}
    Update i-th value network parameters: θ_{V_i} ← θ_{V_i} − β∇_{θ_{V_i}} L_{1:k}[V_{θ_{V_i}}]
end while
```

I SETUP FOR DISCRETE-ACTION ENVIRONMENTS

Load balancing across servers (Figure 1a). In this environment, an RL agent balances jobs over k servers to minimize the average job completion time. We use the same job size distribution and Poisson arrival process as in the example in §3, but run over 10 simulated servers with different processing rates, ranging linearly from 0.15 to 1.05. In each episode, we generate 500 jobs as the exogenous input process. The problem of minimizing average job completion time does not have a closed-form solution (Harchol-Balter & Vesilo, 2010); the most widely-used heuristic is to join the shortest queue (Daley, 1987). However, understanding the workload pattern can give a better policy; for example, we can reserve some servers for small jobs. In this environment, the observed state is a vector of $(j, q_1, q_2, ..., q_k)$, where j is the size of the incoming job, q_i is the amount of work currently in each queue. The action $a \in \{1, 2, ..., k\}$ schedules the incoming job to a specific queue. The reward is the number of active jobs times the negated time elapsed since the last action.

Bitrate adaptation for video streaming (Figure 1b). Streaming video over variable-bandwidth connections requires the client to adapt the video bitrates to optimize the user experience. This is challenging since the available network bandwidth (the exogenous input process) is hard to predict accurately. We simulate real-world video streaming using public cellular network data (Riiser et al., 2013) and video with seven bitrate levels and 500 chunks (DASH Industry Form, 2016). The reward is a weighted combination of video resolution, time paused for rebuffering, and the number of bitrate changes (Mao et al., 2017). The observed state contains bandwidth history, current video buffer size, and current bitrate. The action is the next video chunk's bitrate. State-of-the-art heuristics for this problem conservatively estimate the network bandwidth and use model predictive control to choose the optimal bitrate over the near-term horizon (Yin et al., 2015).

J EXPERIMENT DETAILS

In our discrete-action environments (§6.2), we build 10-value networks and a meta-baseline using MAML (Finn et al., 2017), both on top of the OpenAI A2C implementation (Dhariwal et al., 2017). We use $\gamma = 0.995$ for both environments. The actor and the critic networks have 2 hidden layers, with 64 and 32 hidden neurons on each. The activation function is ReLU (Nair & Hinton, 2010) and the optimizer is Adam (Chilimbi et al., 2014). We train the policy with 16 (synchronous) parallel agents. The learning rate is 1^{-3}. The entropy factor (Mnih et al., 2016) is decayed linearly from 1 to 0.001 over 10000 training iteration. For meta-baseline, the meta learning rate is 1^{-3} and the model specification has 5 step updates, each with learning rate 1^{-4}. The model specification step in MAML is performed with vanilla stochastic gradient descent.

We introduce disturbance into our continuous-action robot control environments (§6.1). For the walker with wind (Figure 1c), we randomly sample a wind force in $[-1, 1]$ initially and add a Gaussian noise sampled from $\mathcal{N}(0, 1)$ at each step. The wind is bounded between $[-10, 10]$. The episode terminates when the walker falls. For the half-cheetah with floating tiles, we extend the number of piers from 10 in the original environment (Clavera et al., 2018) to 50 so that the agent remains on the pathway. We initialize the tiles with damping sampled uniformly in $[0, 10]$. For the 7-DoF robot arm environments, we initialize the target to randomly appear within $(-0.1, -0.2, 0.5), (0.4, 0.2, -0.5)$ in 3D. The position of the target is perturbed with a Gaussian noise sampled from $\mathcal{N}(0, 0.1)$ in each coordinate at each step. We bound the position of the target So that the target is confined within the arm's reach. The episode length of all these environments are capped at 1000.

We build the multi-value networks and meta-baseline on top of the TRPO implementation by OpenAI (Dhariwal et al., 2017). We turned off the GAE enhancement by using $\lambda = 1$ for fair comparison. We found that it makes only a small performance difference (within $\pm 5\%$ using $\lambda = \{0.95, 0.96, 0.97, 0.98, 0.99, 1\}$) in our environments. We use $\gamma = 0.99$ for all three environments. The policy network has 2 hidden layers, with 128 and 64 hidden neurons on each. The activation function is ReLU (Nair & Hinton, 2010). The KL divergence constraint δ is 0.01. The learning rate for value functions is 1^{-3}. The hyperparameter of training the meta-baseline is the same as the discrete-action case.

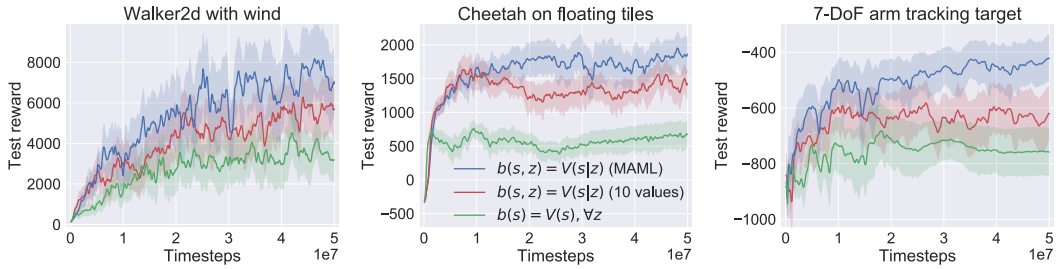

Figure 6: In continuous-action MuJoCo environments (§6.1), PPO (Schulman et al., 2017) with input-dependent baselines achieves 42%–3.5× better testing reward than PPO with a standard state-dependent baseline. Learning curves are on 100 testing episodes with unseen input sequences; shaded area spans one standard deviation.

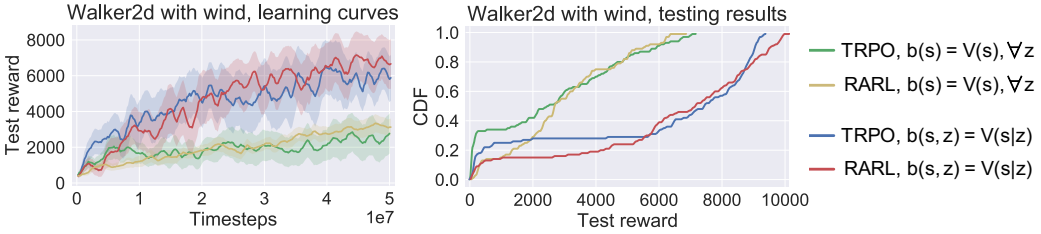

Figure 7: The input-dependent baseline technique is complementary and orthogonal to RARL (Pinto et al., 2017). The implementation of input-dependent baseline is MAML (§5). Left: learning curves of testing rewards; shaded area spans one standard deviation; input-dependent baseline improves the policy optimization for both TRPO and RARL, while RARL improves TRPO in the walker2d environment with wind disturbance. Right: CDF of testing performance; RARL improves the policy especially in the low reward region; applying input-dependent baseline boosts the performance for both TRPO and RARL significantly.

K INPUT-DEPENDENT BASELINES WITH PPO

Figure 6 shows the results of applying input-dependent baselines on PPO (Schulman et al., 2017) in MuJoCo (Todorov et al., 2012) environments. We make three key observations. First, compared to Figure 4 the best performances of PPO in these environments (blue curves) are better than that of TRPO. This is as expected, because the variances of the reward feedbacks in these environments are generally large; and the reward clipping in PPO helps. Second, input-dependent baselines boost the performance on testing rewards for all environments. In particular, meta-learning approach achieves the best performance, as it is not restricted to a fixed set of input sequences during training (§5). Third, the trend of learning curve is similar to that in TRPO (Figure 4), which shows our input-dependent baseline approach is generally applicable to a range of policy gradient based methods (e.g., A2C (§6.2), TRPO (§6.1), PPO).

L INPUT-DEPENDENT BASELINES WITH RARL

Our work is orthogonal and complementary to adversarial and robust reinforcement learning (e.g., RARL (Pinto et al., 2017)). These methods seek to improve policy robustness by co-training an adversary to generate a worst-case noise process, whereas our work improves policy optimization itself in the presence of inputs like noise. Note that if an adversary generates high-variance noise, similar to the inputs we consider in our experiments (§6), techniques such RARL alone are not adequate to train good controllers.

To empirically demonstrate this effect, we repeat the walker2d with wind experiment described in §6.1. In this environment, we add a noise (same scale as the original random walk) on the wind and co-train an adversary to control the strength and direction of this noise. We follow the training procedure described in RARL (Pinto et al., 2017, §3.3).

Figure 7 depicts the results. With either the standard state-dependent baseline or our input-dependent baseline, RARL generally improves the robustness of the policy, as RARL achieves better testing rewards especially in the low reward region (i.e., compared the yellow curve to green curve, or red curve to blue curve in CDF of Figure 7). Moreover, input-dependent baseline significantly improves

the policy optimization, which boosts the performance of both TRPO and RARL (i.e., compared the blue curve to green curve, and the red curve to yellow curve). Therefore, in this environment we empirically show that the input-dependent baseline generally helps improve the policy optimization methods and is complementary to adversarial RL methods such as RARL.