Discriminator Augmented Model-Based Reinforcement Learning

Anonymous Author(s)
Affiliation
Address
email

Abstract

By planning through a learned dynamics model, model-based reinforcement learning (MBRL) offers the prospect of good performance with little environment interaction. However, it is common in practice for the learned model to be inaccurate, impairing planning and leading to poor performance. This paper aims to improve planning with an importance sampling framework that accounts and corrects for discrepancy between the true and learned dynamics. This framework also motivates an alternative objective for fitting the dynamics model: to minimize the variance of value estimation during planning. We derive and implement this objective, which encourages better prediction on trajectories with larger returns. We observe empirically that our approach improves the performance of current MBRL algorithms on two stochastic control problems, and provide a theoretical basis for our method.

1 Introduction

Model free reinforcement learning methods have achieved good performance on a number of complex tasks, but usually require a large amount of data collected through environment interaction [25, 23]. *Model-based* reinforcement learning (MBRL) can potentially reduce these data requirements by fitting a model of the environment dynamics on a small dataset, then planning through the learned dynamics to produce a good policy. However, inaccurate models can severely impair planning performance. Recent techniques attempt to avoid *compounding* model error by restricting the number of model unrolling steps [16, 11], but this creates a trade-off between planning performance and sample efficiency. More importantly, such approaches assume that the learned model mostly matches the true (one timestep) dynamics, and only address compounding error that arises over many timesteps. In this paper we show that there are a range of environments where the learned model can be inaccurate even on short time horizons (e.g., one timestep), so merely mitigating compounding error is insufficient. Given the evidence that even deep neural network models can struggle to learn complex, high dimensional distributions [2], we expect inaccurate learned dynamics to be an increasingly important issue as we apply MBRL to harder and more realistic environments.

Consider a learned dynamics model as a generative model from which we sample transitions. Even if the model is inaccurate, some samples may be better than others. Our approach trains a discriminative model to assess the quality of sampled transitions during planning, and upweight or downweight value estimates computed from high and low quality samples, respectively. In section 3.1 we also show that this method is a form of likelihood-free importance sampling that, assuming the discriminator is an optimal classifier, produces unbiased value estimates for planning. Since our discriminator can correct model error during planning, we are no longer restricted to fitting the dynamics using a maximum likelihood objective. Instead, we can learn *biased* dynamics models with advantageous properties, such as reduced value estimation variance during planning. We derive and implement an

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

objective function for learning these variance-minimizing models. Unlike maximum likelihood, our proposed objective incorporates trajectory return statistics into the model fitting process. We call our planning and model training framework **D**iscriminator **A**ugmented **M**BRL (or **DAM** for short).

To evaluate scenarios where the dynamics is difficult to learn, our experiments consider environments with stochastic and multi-modal dynamics. We find that DAM significantly improves planning performance over existing MBRL algorithms in these environments.

2 Preliminaries

Consider a Markov Decision Process (MDP) $\mathcal{M} \equiv (\mathcal{S}, \mathcal{A}, p, r, \gamma, T)$ [30], where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ denotes the action space, $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto \mathbb{R}_{>0}$ denotes the transition dynamics, $r : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$ denotes the reward function, $\gamma \in [0, 1]$ denotes the discount factor and T denotes the maximum episode horizon. Our objective is to maximize $\mathbb{E}_{p(\tau)}[R(\tau)]$ where $R(\tau) = \sum_{t=1}^{T} \gamma^t r(s_t, a_t)$ denotes the discounted return for the trajectory $\tau = \{s_1, a_1, \dots s_T\}$, which is sampled from

$$p(\tau) = p(s_1) \prod_{t=1}^{T-1} p(s_{t+1}|s_t, a_t)\pi(a_t|s_t). \tag{1}$$

Here, $p : \mathcal{S} \mapsto \mathbb{R}_{>0}$ denotes the initial state distribution, and $\pi(a_t \mid s_t)$ defines our controller which is optimized to maximized the expected discounted return. The overloaded notation p is interpreted based on variable(s) for which the density is computed.

Since p is can only be sampled sequentially in practice, MBRL [20, 28] learns an estimate of the transition dynamics $q : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \mapsto \mathbb{R}_{>0}$. We can also define $q(\tau) = p(s_1) \prod_{t=1}^{T-1} q(s_{t+1}|s_t, a_t)\pi(a_t|s_t)$ analogous to $p(\tau)$. The model is generally learned by maximizing $\mathbb{E}_{s',a,s\sim\mathcal{B}}\big[\log q(s' \mid s, a)\big]$, where $\mathcal{B}$ denotes the dataset of transitions collected in $\mathcal{M}$. The approximate transition dynamics can be used to learn a parametric controller π or perform online planning using sampling based methods.

3 Discriminator Augmented MBRL

Function ESTIMATEVALUE
  input : s_1: Initial state
  input : $\mathbf{a} = \{a_t\}_{t=1}^{H}$: Action sequence
  $w \leftarrow 1$; `// stores weights (Eq. 4)`
  $R \leftarrow 0$;
  for $t \leftarrow 1$ **to** H **do**
    $r_t \leftarrow r(s_t, a_t)$;
    $R \leftarrow R + w \cdot r_t$;
    $s_{t+1} \sim q(\cdot|s_t, a_t)$;
    $w \leftarrow w \cdot \sigma(\tilde{\mathcal{D}}(s_t, a_t, s_{t+1}))$; `// Eq. 8`
  end
  Result: R

Algorithm 1: ESTIMATEVALUE produces an estimated return R of some action sequence using a learned model q. Since q may be biased with respect to the true dynamics, it uses a discriminator $\mathcal{D}$ to correct the estimate as described in Sec. 3.1.

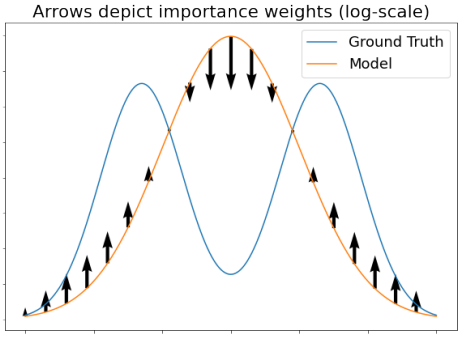

Figure 1: In this illustration, the blue curve depicts a ground truth density and orange a learned model. Importance sampling weights (represented by each arrow's magnitude in log-scale) can help us correct estimates under the incorrect model distribution.

In this work, we look at some of the fundamental components of model-based reinforcement learning. In Section 3.1, we discuss an often ignored discrepancy caused by sampling trajectories from learned dynamics models and discuss how this bias can be corrected using importance sampling. Then, in Section 3.2, we discuss how interpreting the model-based RL in an importance sampling framework suggests a novel objective for learning models for environment dynamics.

3.1 Correcting the Sampling Bias in Model-Based RL

Consider our optimization objective $J(\pi) = \mathbb{E}_{p(\tau)}[R(\tau)]$. Using the approximate dynamics q introduces a bias in our estimation of $J(\pi)$. Model-based RL methods generally use $\tilde{J}_q(\pi) = \mathbb{E}_{q(\tau)}[R(\tau)]$ as a proxy for $J(\pi)$, where the trajectories are sampled from $q(\tau)$ instead of $p(\tau)$ without accounting for the resulting bias. In this work, we mitigate this bias by using the importance sampling framework:

$$\mathbb{E}_{p(\tau)}\left[R(\tau)\right] = \mathbb{E}_{q(\tau)}\left[\frac{p(\tau)}{q(\tau)}R(\tau)\right] = \mathbb{E}_{q(\tau)}\Big[w(\tau)R(\tau)\Big] \tag{2}$$

where we have introduced the importance sampling correction $w(\tau)$ to correct for the bias introduced by sampling trajectories from $q(\tau)$. Simplifying $w(\tau)$, we get

$$w(\tau) = \prod_{t=1}^{T-1} \frac{p(s_{t+1} \mid s_t, a_t)}{q(s_{t+1} \mid s_t, a_t)} = \prod_{t=1}^{T-1} w(s_{t+1}, a_t, s_t) \tag{3}$$

By exploiting the MDP's temporal structure, we can obtain an improved importance sampling estimator that weights the per-timestep rewards instead of the trajectory return. Comparing to Eq. 2, we see that this estimator multiplies the reward at time t by a weight that only depends on transitions leading up to it, and not transitions that occur after it:

$$\mathbb{E}_{p(\tau)}[R(\tau)] = \sum_{t=1}^{T} \mathbb{E}_q \left[\left(\prod_{m=1}^{t} w(s_m, a_m, s_{m+1}) \right) r(s_t, a_t) \right] \tag{4}$$

As stated earlier, we only have access to the samples from the transition dynamics and not the probabilities. Thus to compute $w(\tau)$ for $\tau \sim q$, we use techniques from density ratio estimation [36, 13, 33]. We setup a binary classification problem over $\mathcal{D} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \times \mathcal{Y} \mapsto [0, 1]$, where $\mathcal{Y} = \{0, 1\}$. Using the transition samples from the environment and the learned dynamics, we create a dataset of transitions $\{s', a, s, y\}$, where $y = 1$ for $(s', a, s) \sim p(s' \mid s, a)p(s, a)$ and $y = 0$ for $(s', a, s) \sim q(s' \mid s, a)p(s, a)$. Effectively, we are training a discriminator to distinguish between real transitions in the environment and the transitions sampled from the learned model. Alg. 2 provides the pseudocode for training the discriminator.

To connect the discriminator to the importance weights, consider the Bayes-optimal classifier for this problem:

$$\mathcal{D}(s', a, s) = \frac{p(s' \mid s, a)p(s, a)}{p(s' \mid s, a)p(s, a) + q(s' \mid s, a)p(s, a)} \tag{5}$$

$$= \frac{p(s' \mid s, a)/q(s' \mid s, a)}{p(s' \mid s, a)/q(s' \mid s, a) + 1} \tag{6}$$

$$= \sigma\Big(\log \frac{p(s' \mid s, a)}{q(s' \mid s, a)}\Big) = \sigma\Big(\tilde{\mathcal{D}}(s', a, s)\Big) \tag{7}$$

$\tilde{\mathcal{D}}$ is the classifier logits, and we can recover the importance weight by exponentiating them:

$$w(s', a, s) = \frac{p(s', a, s)}{q(s', a, s)} = \exp\left(\tilde{\mathcal{D}}(s', a, s)\right) \tag{8}$$

For our transitions $\{s', a, s\}$, it is important to sample $s, a \sim p(s, a)$ (that is, real state-action pairs) so that the logits can be connected to the importance weights. Since, we only need samples to train the classifier, this allows us to estimate the importance weights without requiring probabilities from underlying transition dynamics. Alg. 1 provides the pseudocode for estimating the discriminator-corrected return of an action sequence, which forms the subroutine for choosing actions in sampling based planning algorithms.

The benefit of this approach arises from the potential simplicity of the discriminative problem compared to the modelling the dynamics. The dynamics can be hard to model in the real world, as they often involve higher dimensional spaces (like RGB images) and can be multi-modal due to partial observability. The modelling and optimization challenges can often result in an incorrectly learned model, an example of which is shown in Fig 1. On the other hand, discrimination can be

an easier problem as fake and real samples can be well separated, especially in higher-dimensional spaces. This can improve the value estimation by down-weighting samples which do not fit with the real world samples. If the learned dynamics are indeed correct, the true and fake samples will be indistinguishable and the importance sampling weights will be close to unity, reducing the problem to conventional model-based RL.

3.2 Minimum Variance Dynamics Model Fitting

The above discussion applies to any learned model, including the conventional approach of learning by maximizing the log-likelihood of transitions collected in the environment. The policy objective under q, that is $\hat{J}_q(\pi) = \frac{1}{N}\sum_{i=1}^{N} w(\tau_i)R(\tau_i)$, is an unbiased estimator of $J(\pi)\ \forall q$ as long as the support of p is a subset of support of q. However, from an optimization perspective, it is desirable to have an estimator with the minimum variance. Sampling based planning algorithms generate candidate action sequences and rank them according to their value function. Minimizing the variance of the value estimate of an action sequence increases the likelihood of the action sequences getting ranked correctly, and thus, the best action sequence getting chosen. Thus, it is favorable to reduce the variance of our estimate $\hat{J}_q$. Assuming that the $R(\tau) \geq 0$, it can be shown that

$$q^*(\tau) = \frac{R(\tau)p(\tau)}{\mathbb{E}_{p(\tau)}\left[R(\tau)\right]} \tag{9}$$

Here, q^* denotes the sampling distribution which minimizes the variance of $\hat{J}_q$. While the denominator for the optimal sampling distribution is intractable, the important distinction here is that trajectories should be sampled in accordance to their density under the true trajectory distribution and the return accumulated by the trajectory. Using the q^* as the target distribution, we can setup our minimization objective for model-based learning to be $\mathrm{KL}(q^*(\tau)\ ||\ q(\tau))$. As we show in Appendix A.1, the gradient with respect to q is:

$$\nabla_q \mathrm{KL}(q^*(\tau)\ ||\ q(\tau)) \propto -\sum_{t=1}^{T-1} \mathbb{E}_{p(\tau)}\left[R(\tau)\nabla_q \log q(s_{t+1}\mid s_t, a_t)\right] \tag{10}$$

Intuitively, this corresponds to upweighting the gradient for trajectories with higher return, forcing the dynamics model to fit better to transitions with higher return. In particular, this highlights that the conventional approach of learning dynamics does not represent the optimal approach in the importance sampling framework. The modified model training algorithm is described in Alg. 2 in the appendix.

3.3 Planning and Training Loop

The previous two sections describe how to train the dynamics model, and how to train a discriminator that corrects the sampling bias from that learned model. Further, ESTIMATEVALUE (Alg. 1) shows how to implement discriminator-corrected value estimation. We can plug ESTIMATEVALUE as a subroutine into most sampling based planners. Once a planner is in place, we can use it execute actions in the real environment, collect more data, and re-train our model and discriminator. In Appendix B we discuss in detail the implementation of this model-based RL loop within our model and discriminator training framework.

4 Experiments

In this section we evaluate our proposed approach and compare with model-based baselines. We hypothesize that environments with multi-modal dynamics make model fitting difficult, which would pose problems for planning. Indeed, we observe that our proposed approach improves performance over standard model-based RL in such settings.

We devise and implement two environments with stochastic, multi-modal dynamics: **IcyRoad** and **Intersection**. In **IcyRoad** the agent drives a car along an icy road, and above a threshold speed the car has a probability of swerving off the road in either direction (hence the multi-modality) and incurring a large negative reward. In **Intersection** the agent drives a car into an intersection. An oncoming car is also entering the intersection and will turn either left or right with equal probability–the outcome is

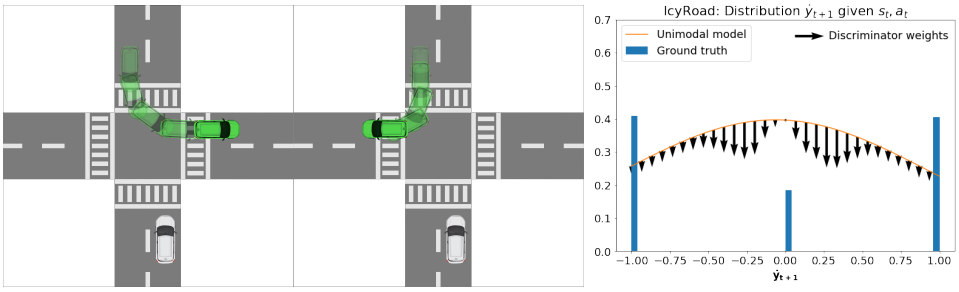

Figure 2: **Left**: Depiction of different scenarios in the Intersection environment, where the agent controls the silver car. Both cars are entering the intersection at the same time, but the agent does not know whether the oncoming green car will turn left (left panel) or turn right (right panel). If the agent drives into the intersection too fast, it may collide. **Right**: The learned (unimodal) model and ground truth distributions $p(\dot{y}_{s+1}|s_t, a_t)$, where $s_t = (2, 0, 2, 0)$ and the action is to accelerate. After training with maximum likelihood, the unimodal model erroneously places large probability mass on unrealistic regions of the state space. The discriminator produces importance weights (black arrows, log scale) that downweight the model's samples in the unrealistic regions.

Average return	IcyRoad		Intersection	
	Unimodal	Multimodal	Unimodal	Multimodal
Baseline	0.71 ± 1.72	5.34 ± 0.84	62.67 ± 22.10	29.58 ± 0.105
DAM (minvar only)	0.86 ± 0.79	4.55 ± 3.50	31.40 ± 1.111	31.30 ± 1.634
DAM (disc only)	4.88 ± 0.50	3.30 ± 2.75	87.79 ± 15.34	60.18 ± 19.58
DAM	2.32 ± 1.76	$\mathbf{7.94 \pm 0.92}$	46.76 ± 9.543	$\mathbf{97.72 \pm 5.127}$

Table 1: Average return for each method in the IcyRoad and Intersection environments. Columns are split by whether they use a unimodal (Gaussian) or multimodal (mixture density network) learned dynamics model. Combining our method **DAM** with multimodal learned dynamics models usually performs best among different methods we evaluated. Results show mean and standard error across 3 runs (random seeds) for each hyperparameter setting.

unknown in advance to the agent. Fig. 2 illustrates the Intersection environment, and Appendix C both environments in more detail.

For our baseline, we fit a dynamics model using the usual maximum likelihood loss and use a standard random shooting planner with uncorrected value estimates. In **DAM** we use both aspects of our method: we train the model using the minimum variance objective (Eq. 10) and plan using the discriminator-corrected estimator (Eq. 4). To disentangle the effects of each component of our method, we also evaluate variants that use only one aspect of our method, but not both. **DAM (disc only)** only does discriminator-corrected planning, while **DAM (minvar only)** only uses the minimum variance object. Practitioners typically train models that output the parameters for a Gaussian distribution, which is unimodal. Since our environment dynamics are multimodal, it also makes sense to try fitting multimodal models such as mixture density networks [4]. We test each of our planning and training methods with both a mixture density network model and a standard Gaussian model, and display both results.

Table 1 depicts the results after 10 rounds of interleaving model (+ discriminator) training with data collection in the environment. We see that in both environments, **DAM** using multimodal dynamics models obtains the best results. **DAM (minvar only)**, which only uses the minimum variance objective without discriminator-corrected planning, is often *worse* than the baseline. We hypothesize this occurs because the minimum variance objective increases model bias versus standard maximum likelihood training, which can be harmful without discriminator-corrected planning. When using unimodal models in either environment, we see that simply having a discriminator for planning already increases performance over the baseline. Fig. 2 depicts our analysis of what the trained model and discriminator are doing in IcyRoad: we see that the unimodal model is struggling to fit the true dynamics, while the discriminator is helping to correct the model error. Interestingly, adding the minimum variance objective to discriminator-corrected planning does *not* help with unimodal models, but it *does* help with multimodal models.

5 Related Work

Model-based RL has a rich history in control and robotics. Classically, the methods for model-based control have assumed access to underlying dynamics [35, 22, 26, 24] or make simplifying assumptions about the environment dynamics [7, 8, 21, 18]. While such simplifying assumptions can yield exact or efficient controllers, the models cannot faithfully represent the underlying dynamics, especially when considering high dimensional state spaces like RGB images. Recent work has focused on combining high-capacity function approximators such as neural networks with model-based reinforcement learning [20, 27, 28, 6, 12, 32, 14, 29], which employ sampling-based planning methods [38, 27, 31]. Concurrently, some recent work has integrated policy networks with deep models [16, 37] in spirit of Dyna [34]. Neural networks allow more flexible representation of non-linear dynamics, allowing model-based RL to scale to complex high dimensional tasks [17, 28]. However, despite the incredible progress, none of these prior works account for the bias introduced by using learned models for planning.

Classically, the field of robust control has looked at designing controllers for with uncertain transition models [3, 9]. More relevantly, [1] incorporates bias correction for approximate transition models to learn better policies. However, they only focus on approximation error which arises due to the use of simple linear models. More recently, [19] discusses the mismatch which arises between the training objectives and the control objectives, and how that discrepancy can lead to biases in model learning. In contrast to these works, our work focuses on the sampling bias introduced despite using flexible function approximators like neural networks. To some extent, our work also addresses the bias discussed in [19], as our objective encourages the transition model to fit better to trajectories with higher return.

Prior work has considered estimating value functions using transition models [15, 11, 5]. However, our work addresses the biased values generated by the use of learned transition models. To that extent, we rely on classifier-based density ratio estimation techniques [33], which have been employed in generative modelling [36, 13]. [10] use similar principles to correct for differences in dynamics when transferring experience between environments, where the classifier's corrections are realized by adjusting the reward appropriately before applying model-free RL. To the best of our knowledge, ours is first work which raises and addresses the issue of sampling bias in context of model-based RL.

6 Conclusion

We introduce a framework for mitigating the impact of model error on planning in model-based RL. Our framework trains discriminators to correct value estimation bias, while also introducing a novel model training objective for minimizing value estimation variance during planning. Empirically, we find that these modifications improve agent performance in environments with stochastic and multi-modal dynamics that pose a challenge for standard model fitting techniques. By improving planning in environments that are difficult to model properly, this framework is a step towards more efficient and performant reinforcement learning in complex, real world problems.

References

[1] P. Abbeel, M. Quigley, and A. Y. Ng. Using inaccurate models in reinforcement learning. In *Proceedings of the 23rd international conference on Machine learning*, pages 1–8, 2006.

[2] S. Arora, A. Risteski, and Y. Zhang. Do gans learn the distribution? some theory and empirics. In *International Conference on Learning Representations*, 2018.

[3] J. A. Bagnell, A. Y. Ng, and J. G. Schneider. Solving uncertain markov decision processes. 2001.

[4] C. M. Bishop. Mixture density networks. 1994.

[5] J. Buckman, D. Hafner, G. Tucker, E. Brevdo, and H. Lee. Sample-efficient reinforcement learning with stochastic ensemble value expansion. In *Advances in Neural Information Processing Systems*, pages 8224–8234, 2018.

[6] K. Chua, R. Calandra, R. McAllister, and S. Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*, pages 4754–4765, 2018.

[7] M. Deisenroth and C. E. Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pages 465–472, 2011.

[8] M. P. Deisenroth, D. Fox, and C. E. Rasmussen. Gaussian processes for data-efficient learning in robotics and control. *IEEE transactions on pattern analysis and machine intelligence*, 37(2):408–423, 2013.

[9] L. El Ghaoui and A. Nilim. Robust solutions to markov decision problems with uncertain transition matrices. *Operations Research*, 53(5):780–798, 2005.

[10] B. Eysenbach, S. Asawa, S. Chaudhari, R. Salakhutdinov, and S. Levine. Off-dynamics reinforcement learning: Training for transfer with domain classifiers. *arXiv preprint arXiv:2006.13916*, 2020.

[11] V. Feinberg, A. Wan, I. Stoica, M. I. Jordan, J. E. Gonzalez, and S. Levine. Model-based value estimation for efficient model-free reinforcement learning. *arXiv preprint arXiv:1803.00101*, 2018.

[12] Y. Gal, R. McAllister, and C. E. Rasmussen. Improving pilco with bayesian neural network dynamics models. In *Data-Efficient Machine Learning workshop, ICML*, volume 4, page 34, 2016.

[13] A. Grover, J. Song, A. Kapoor, K. Tran, A. Agarwal, E. J. Horvitz, and S. Ermon. Bias correction of learned generative models using likelihood-free importance weighting. In *Advances in Neural Information Processing Systems*, pages 11058–11070, 2019.

[14] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning*, pages 2555–2565. PMLR, 2019.

[15] N. Heess, G. Wayne, D. Silver, T. Lillicrap, T. Erez, and Y. Tassa. Learning continuous control policies by stochastic value gradients. In *Advances in Neural Information Processing Systems*, pages 2944–2952, 2015.

[16] M. Janner, J. Fu, M. Zhang, and S. Levine. When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems*, pages 12519–12530, 2019.

[17] L. Kaiser, M. Babaeizadeh, P. Milos, B. Osinski, R. H. Campbell, K. Czechowski, D. Erhan, C. Finn, P. Kozakowski, S. Levine, et al. Model-based reinforcement learning for atari. *arXiv preprint arXiv:1903.00374*, 2019.

[18] V. Kumar, E. Todorov, and S. Levine. Optimal control with learned local models: Application to dexterous manipulation. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 378–383. IEEE, 2016.

[19] N. Lambert, B. Amos, O. Yadan, and R. Calandra. Objective mismatch in model-based reinforcement learning. *arXiv preprint arXiv:2002.04523*, 2020.

[20] E. Langlois, S. Zhang, G. Zhang, P. Abbeel, and J. Ba. Benchmarking model-based reinforcement learning. *arXiv preprint arXiv:1907.02057*, 2019.

[21] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016.

[22] W. Li and E. Todorov. Iterative linear quadratic regulator design for nonlinear biological movement systems. In *ICINCO (1)*, pages 222–229, 2004.

[23] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[24] K. Lowrey, A. Rajeswaran, S. Kakade, E. Todorov, and I. Mordatch. Plan online, learn offline: Efficient learning and exploration via model-based control. *arXiv preprint arXiv:1811.01848*, 2018.

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

[26] I. Mordatch, E. Todorov, and Z. Popović. Discovery of complex behaviors through contact-invariant optimization. *ACM Transactions on Graphics (TOG)*, 31(4):1–8, 2012.

[27] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7559–7566. IEEE, 2018.

[28] A. Nagabandi, K. Konolige, S. Levine, and V. Kumar. Deep dynamics models for learning dexterous manipulation. In *Conference on Robot Learning*, pages 1101–1112, 2020.

[29] S. Nair, S. Savarese, and C. Finn. Goal-aware prediction: Learning to model what matters. *arXiv preprint arXiv:2007.07170*, 2020.

[30] M. L. Puterman. Markov decision processes. *Handbooks in operations research and management science*, 2:331–434, 1990.

[31] R. Y. Rubinstein and D. P. Kroese. *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning*. Springer Science & Business Media, 2013.

[32] A. Sharma, S. Gu, S. Levine, V. Kumar, and K. Hausman. Dynamics-aware unsupervised discovery of skills. *arXiv preprint arXiv:1907.01657*, 2019.

[33] M. Sugiyama, T. Suzuki, and T. Kanamori. *Density ratio estimation in machine learning*. Cambridge University Press, 2012.

[34] R. S. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Machine learning proceedings 1990*, pages 216–224. Elsevier, 1990.

[35] Y. Tassa, T. Erez, and E. Todorov. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4906–4913. IEEE, 2012.

[36] D. Tran, R. Ranganath, and D. Blei. Hierarchical implicit models and likelihood-free variational inference. In *Advances in Neural Information Processing Systems*, pages 5523–5533, 2017.

[37] T. Wang and J. Ba. Exploring model-based planning with policy networks. *arXiv preprint arXiv:1906.08649*, 2019.

[38] G. Williams, A. Aldrich, and E. Theodorou. Model predictive path integral control using covariance variable importance sampling. *arXiv preprint arXiv:1509.01149*, 2015.