Boney, Rinu; Kannala, Juho; Ilin, Alexander

Regularizing Model-Based Planning with Energy-Based Models

Published in:
Proceedings of the Conference on Robot Learning

Published: 01/01/2020

Document Version
Publisher's PDF, also known as Version of record

Please cite the original version:
Boney, R., Kannala, J., & Ilin, A. (2020). Regularizing Model-Based Planning with Energy-Based Models. In *Proceedings of the Conference on Robot Learning* (pp. 182-191). (Proceedings of Machine Learning Research ; Vol. 100). JMLR. http://proceedings.mlr.press/v100/boney20a.html

Regularizing Model-Based Planning with Energy-Based Models

Rinu Boney **Juho Kannala** **Alexander Ilin**
Department of Computer Science
Aalto University, Finland
firstname.lastname@aalto.fi

Abstract: Model-based reinforcement learning could enable sample-efficient learning by quickly acquiring rich knowledge about the world and using it to improve behaviour without additional data. Learned dynamics models can be directly used for planning actions but this has been challenging because of inaccuracies in the learned models. In this paper, we focus on planning with learned dynamics models and propose to regularize it using energy estimates of state transitions in the environment. We visually demonstrate the effectiveness of the proposed method and show that off-policy training of an energy estimator can be effectively used to regularize planning with pre-trained dynamics models. Further, we demonstrate that the proposed method enables sample-efficient learning to achieve competitive performance in challenging continuous control tasks such as Half-cheetah and Ant in just a few minutes of experience.

Keywords: model-based reinforcement learning, energy-based learning

1 Introduction

Reinforcement learning deals with taking sequences of actions in previously unknown environments, to maximize cumulative rewards. Deep reinforcement learning combines the power of deep neural networks for function approximation with reinforcement learning algorithms. Such algorithms have recently been successful in solving several challenging problems [1, 2, 3]. However, these algorithms are often not sample-efficient, typically requiring an unreasonable amount of interactions in an environment [1, 3, 4]. In real world applications, it is often the case that collecting data is expensive, motivating the need for algorithms that can learn from a minimal number of interactions with the environment.

Effectively using a model of the environment for learning could result in a significant increase of sample-efficiency [5, 6]. This is because models introduce an inductive bias that the environment behaves in a forward generative causal direction. Models can be directly used for planning actions that maximize the sum of expected rewards in a finite horizon. Previous works have successfully used fast and accurate simulators of the environment for planning, to solve challenging tasks such as control of a Humanoid [6, 7]. However, most real-world problems do not come with a fast and accurate simulator and engineering one from scratch is a laborious task. Even if a simulator exists: 1) maintaining the simulator to accommodate for changes in the real-world is a lot of manual work, and 2) matching the true state of the environment and the simulator is not trivial. This motivates the need for *learning* dynamics models of the environment. Dynamics models of the environment can be efficiently learned using supervised learning techniques [8] and also efficiently adapted to changes in the environment [9].

Planning with learned dynamics models is challenging because the planner can exploit the inaccuracies of the model to arrive at actions that are imagined to produce highly over-optimistic rewards (see Figure 5). In this paper, we propose to regularize model-based planning with an energy-based model trained on the same transitions as the dynamics model. The planning procedure is augmented with an additive cost of minimizing the energy of the imagined transitions. This penalizes the planner from producing trajectories with transitions that are outside the training data distribution (that

3rd Conference on Robot Learning (CoRL 2019), Osaka, Japan.

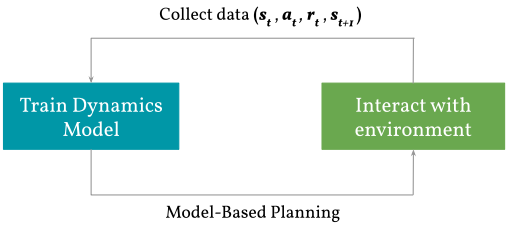

Figure 1: Overview of training loop. We initially perform random exploration for one or more episodes to train the dynamics model and then interact with the environment by planning using the learned dynamics model. At the end of each episode, we re-train the dynamics model on the past experience.

is, transitions with high energy estimates). We demonstrate that the proposed method is effective at regularizing model-based planning from exploiting inaccuracies of the model. Previous works have proposed to use ensembles of models to deal with this problem [5]. We show that the proposed method can further improve the performance of planning on top of pre-trained ensemble of models. Furthermore, we show that the proposed method enables sample-efficient learning to achieve competitive performance in five popular continuous control tasks.

2 Model-Based Planning

In this section, we formalize the problem setting as a Markov Decision Process (MDP). At every discrete time-step t, the environment is in state s_t, the agent takes an action a_t to receive a scalar reward $r_t = r(s_t, a_t)$ and the environment transitions to the next state s_{t+1}, following the dynamics $s_{t+1} = f(s_t, a_t)$. The goal of the agent is to choose actions a_t so as to maximize the sum of expected future rewards (called return), $G = E\left[\sum_{t=0}^{\infty} r(s_t, a_t)\right]$.

In this paper, we focus on finite-horizon planning with learned dynamics models $\hat{f}$. Also, we assume that the reward function $r(s_t, a_t)$ is known and that the state s_t is fully observed. Forward dynamics models of the environment can be learned using supervised learning techniques to predict the next state s_{t+1} from the current state s_t and action a_t:

$$s_{t+1} = \hat{f}(s_t, a_t)\,.$$

At time-step t, the agent can plan a sequence of actions $\{a_t, \ldots, a_{t+H}\}$ by unrolling the learned dynamics model to maximize the sum of rewards:

$$a_t^*, \ldots, a_{t+H}^* = \underset{a_t, \ldots, a_{t+H}}{\operatorname{argmax}}\; E\left[\sum_{\tau=t}^{t+H} r(s_\tau, a_\tau)\right]\,, \tag{1}$$

such that $s_{\tau+1} = f(s_\tau, a_\tau)$. Since we do not have access to the true dynamics f, we plan using the following objective as a proxy to the true objective:

$$a_t^*, \ldots, a_{t+H}^* = \underset{a_t, \ldots, a_{t+H}}{\operatorname{argmax}} \sum_{\tau=t}^{t+H} r(s_\tau, a_\tau)\,, \tag{2}$$

such that $s_{\tau+1} = \hat{f}(s_\tau, a_\tau)$. We use model-predictive control (MPC) [10, 8] to adapt our plans to new states, that is we apply just the first action from the optimized sequence and re-plan at next step. This reduces the effect of error accumulation due to multi-step predictions using the model.

We initially train the dynamics model using data collected by executing a random policy for one or more episodes. More episodes of initial exploration reduces the effect of the initial model weights. The initial data can also be collected by an existing policy, for example human demonstrations or human-engineered controllers. After training the dynamics models, we interact with the environment using model-based planning (Equation 2). The (s_t, a_t, s_{t+1}) transition observed at each interaction is stored in a replay buffer along with the initial data and the model is re-trained on the replay buffer at the end of each episode. We iterate this alternating process of training the dynamics model and model-based planning. An overview of the training loop is illustrated in Figure 1.

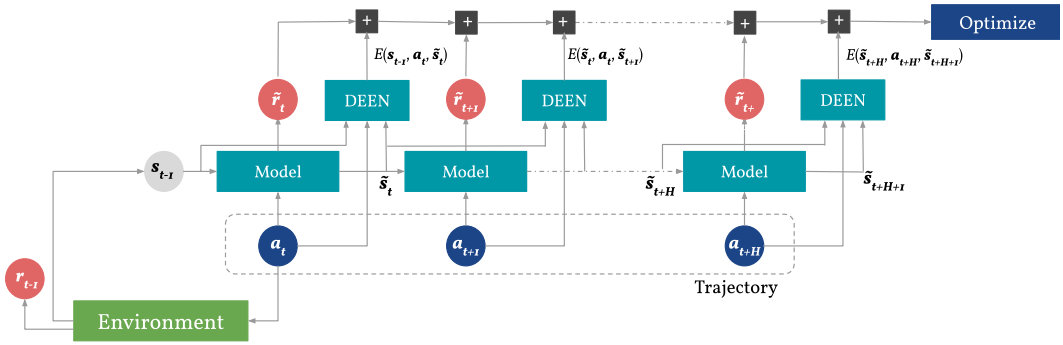

Figure 2: Computational graph of regularizing model-based planning with energy-based models. At each timestep $t-1$, the environment is in a state s_{t-1} and we initially consider an action trajectory $\{a_t, a_{t+1}, \ldots, a_{t+H}\}$ that we could apply for the next H timesteps. The dynamics model is used to predict the future state transitions $\{\tilde{s}_t, \tilde{s}_{t+1}, \ldots, \tilde{s}_{t+H}\}$ at each timestep for the considered action trajectory. The rewards at each time-step is computed directly from the state-action pairs, resulting in a prediction of the finite-horizon cumulative reward. An additive regularization term is augmented to the planning objective by computing the energy of each $(\tilde{s}_\tau, a_\tau, \tilde{s}_{\tau+1})$ transition. The action trajectory $\{a_t, a_{t+1}, \ldots, a_{t+H}\}$ can be optimized to maximize this regularized planning objective.

2.1 Regularizing Model-Based Planning

Directly optimizing the objective in Equation 2 is challenging because $\hat{f}$ is only an approximation of the true dynamics f. Deep neural networks are commonly used as function approximators to learn $\hat{f}$ and they are amenable to erroneous predictions for samples outside the data distribution. An effective optimizer can be easily deceived by these erroneous predictions, converging to action trajectories that are imagined to produce very high rewards but is not the case in reality. This problem can be alleviated by penalizing the optimizer from considering trajectories that are outside the training distribution of the learned dynamics model $\hat{f}$. This can be achieved by augmenting the planning objective with an additive term to maximize the probability of imagined transitions:

$$a_t^*, \ldots, a_{t+H}^* = \underset{a_t, \ldots, a_{t+H}}{\operatorname{argmax}} \sum_{\tau=t}^{t+H} r(s_\tau, a_\tau) + \alpha \log p(s_t, a_t, s_{t+1}, \ldots, s_{t+H}, a_{t+H}, s_{t+H+1}) \,.$$

where the scalar α modulates the weight between both costs. We approximate the joint probability of the whole trajectory as a sum of joint probabilities of each transition in the trajectory:

$$a_t^*, \ldots, a_{t+H}^* = \underset{a_t, \ldots, a_{t+H}}{\operatorname{argmax}} \sum_{\tau=t}^{t+H} \left[r(s_\tau, a_\tau) + \alpha \log p(s_\tau, a_\tau, s_{\tau+1}) \right] \,. \tag{3}$$

Assume that we want to learn the probability density function using a parameterized model $p(x_\tau; \theta)$, where $x_\tau = [s_\tau, a_\tau, s_{\tau+1}]$. The energy function $E(x_\tau; \theta)$ is the unnormalized log-density, that is the probability density function $p(x_\tau)$ is defined as:

$$p(x_\tau; \theta) = \frac{1}{Z(\theta)} \exp\left(-E(x_\tau; \theta)\right),$$

where $Z(\theta) = \int \exp\left(-E(x'_\tau; \theta)\right) dx'_\tau$ is the partition function which normalizes the probability density. Computing the partition function is generally intractable in practice and is not important for regularization in Equation 3 since it does not depend on x_τ. We can instead learn and use the energy function for regularizing model-based planning:

$$a_t^*, \ldots, a_{t+H}^* = \underset{a_t, \ldots, a_{t+H}}{\operatorname{argmax}} \sum_{\tau=t}^{t+H} \left[r(s_\tau, a_\tau) - \alpha E(s_\tau, a_\tau, s_{\tau+1}) \right] \,. \tag{4}$$

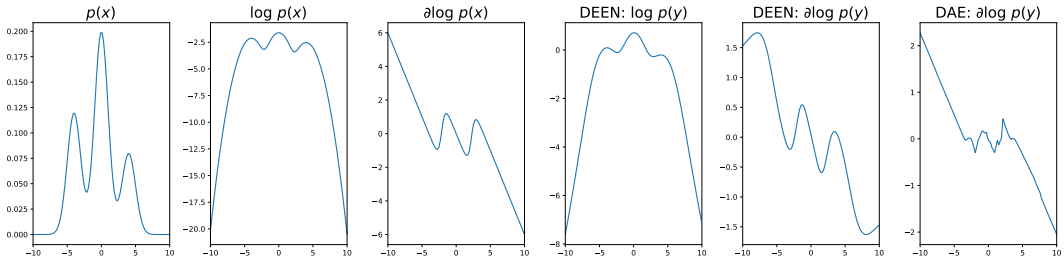

Figure 3: Comparison of score function estimation using DEEN vs DAE. We consider a simple mixture of 1d Gaussian distributions and generate 1000 samples from it. The ground truth probability density function $p(x)$, $\log\ p(x)$ and score function $\frac{\partial \log\ p(x)}{\partial x}$ are shown in the first three figures. We corrupt the training data using additive Gaussian noise with a scale of 0.5. We train a DEEN and a DAE on this training data and the energy and score function estimates of the corrupted distribution $p(y)$ are shown in the last three figures. DEEN provides good smooth estimates of the energy and score function. DAE also provides a reasonable estimate of the score function.

3 Energy-Based Models

In principle, any energy-based model [11] can be used to estimate $E(s_\tau, a_\tau, s_{\tau+1})$ in Equation 4. In this paper, we use the recently introduced Deep Energy Estimator Networks (DEEN) [12, 13] for energy estimation. In this section, we introduce deep energy estimator networks and further contrast it against direct score function estimation using a denoising autoencoder. We show that deep energy estimator networks offers a principled and scalable way to estimate both the energy and score functions, making it a good choice for regularization in both gradient-free and gradient-based planning.

Consider a random variable Y that is a noisy observation of another unknown random variable X with density function $p(x)$. Robbins [14] derived the least squares estimators of variable X from observed value y of variable Y for Poisson, geometric and binomial noise distributions and Miyasawa [15] later extended the work to derive the estimator for univariate Gaussian noise. Raphan and Simoncelli [16] generalized these results into a unified framework and derived least square estimators for more distributions including multivariate Gaussian distribution. The empirical Bayes least squares estimator or the optimal denoising function $g(y)$ for zero-mean multivariate Gaussian noise is given by:

$$g(y) = y + \sigma^2 \nabla_y \log p(y)\,, \tag{5}$$

where $y \sim x + N(0, \sigma^2 I_d)$. Assume that we have access to samples $x_i \in X$. Then, we can corrupt the samples using zero-mean Gaussian noise to obtain samples $y_i \in Y$. We can train a feedforward neural network $\hat{g}$ to denoise each sample y_i to predict x_i. Such a function $\hat{g}$ can be implemented with a denoising autoencoder (DAE) and based on Equation 5 we can use it to approximate the score function $\nabla_y \log p(y)$ of the corrupted distribution as follows:

$$\nabla_y \log p(y) \propto \hat{g}(y) - y\,. \tag{6}$$

Boney et al. [17] proposed to use this approximation to directly estimate the gradient of $\log p(s_\tau, a_\tau, s_{\tau+1})$ in Equation 3. This can be used in gradient-based planning by using the penalty term $\|\hat{g}(x) - x\|^2$ instead of $\log p(s_\tau, a_\tau, s_{\tau+1})$ in Equation 3 and stopping the gradient propagation through the denoising autoencoder g.

Gradient-free planning with the regularized objective in Equation 4 requires explicit energy estimates. Boney et al. [17] used DAE regularization for gradient-free planning, which is not accurate as the denoising error does not correspond to energy estimation. Gradient-free planning offers some attractive properties: 1) It is very easy to parallelize, 2) There is no need to backpropogate gradients, 3) Gradient-based planning involves backpropagating through very deep networks (where the same network is repeatedly applied for a finite horizon), where the problem of exploding and vanishing gradients may arise, and 4) The learned dynamics model can become chaotic leading to high variance in the backpropagated gradients [18]. In this paper, we propose to use differentiable energy-based models to obtain explicit energy estimates, from which the score function can be computed (if needed).

Saremi et al. [12] proposed to explicitly parameterize the energy function $E(y;\theta)$ with a neural network and to compute the derivative of the energy by backpropagating through the network. Such a network can be trained by minimizing the following objective based on the relation in Equation 5:

$$\underset{\theta}{\operatorname{argmin}} \sum_{x_i \in X, y_i \in Y} \left\| x_i - y_i + \sigma^2 \frac{\partial E(y = y_i; \theta)}{\partial y} \right\|^2 \tag{7}$$

The energy function network $E(y;\theta)$ is called deep energy estimator network (DEEN). Note that minimizing this objective involves double backpropagation at each step. Optimizing the objective in Equation 7 ensures that the score function $\partial E(y;\theta)/\partial y$ satisfies the relation in Equation 5. This leads the energy network E to explicitly learn the energy function of the corrupted distribution such that the gradient of the network also corresponds to the score function of the corrupted distribution. In this paper, we propose to use the energy network $E(y;\theta)$ to learn the energy function $E(s_\tau, a_\tau, s_{\tau+1})$ and use it for regularization in the planning objective (Equation 4). A computational graph of regularizing model-based planning with energy estimation using a DEEN network is illustrated in Figure 2.

It is to be noted that both the denoising autocoder and the DEEN methods approximate the score function of the corrupted distribution $p(y)$ instead of the true data distribution $p(x)$. This can potentially behave better in practice since $p(y)$ can be seen as a Parzen window estimate of $p(x)$ with variance σ^2 as the smoothing parameter [13, 19].

In Section 4, we show that energy estimation with DEEN is more effective than direct score function estimation using DAE for regularizing model-based planning. Deep energy estimator networks have been shown to be robust for score function estimation since the score function is computed from explicit energy estimates [12]. We compare score function estimation using denoising autoencoders and deep energy estimator networks on a toy example in Figure 3. Previous works [13, 20] have also observed that directly estimating the score function is not robust in practice.

4 Experiments

We compare the proposed energy-based regularization method to probabilistic ensembles with trajectory sampling (PETS) [5] and DAE regularization [17]. PETS is a state-of-the-art model-based algorithm which involves learning an ensemble of probabilistic dynamics models. We perform the comparison in five popular continuous control benchmarks from [21]: Cartpole, Reacher, Pusher, Half-cheetah and Ant. We use cross-entropy method (CEM) [22] as the optimizer in all our experiments since it is computationally significantly faster than the Adam optimizer used in [17] and also was able to achieve competitive or even better results in these benchmarks. In Section 4.1, we test energy-based regularization method on top of the pre-trained PETS models to show that energy-based regularization further improves planning. In Section 4.2, we show that enery-based regularization enables sample-efficient learning to solve all tasks from just a handful of trials.

4.1 Experiments on Pre-Trained Dynamics Models

We test the proposed regularization on top of the state-of-the-art model-based RL algorithm: PETS. We trained an ensemble of probabilistic dynamics models using the code provided by the authors of [5]. The results are shown in Table 1. We trained PETS on the Half-cheetah benchmark for 300 episodes and perform closed-loop planning by augmenting the planning objective with an additive term consisting of the energy estimates (Equation 4). Both DAE and DEEN regularization are able to improve upon PETS, with DEEN regularization performing the best. Similar to [17], we did not observe any improvements on tasks with low-dimensional action spaces: Cartpole, Reacher and Pusher.

Table 1: Comparison of planning using pre-trained PETS models with different optimizers

Optimizer	CEM	CEM + DAE	Adam + DAE	CEM + DEEN
Return	10955 ± 2865	12967 ± 3216	12796 ± 2716	$\mathbf{13052 \pm 2814}$

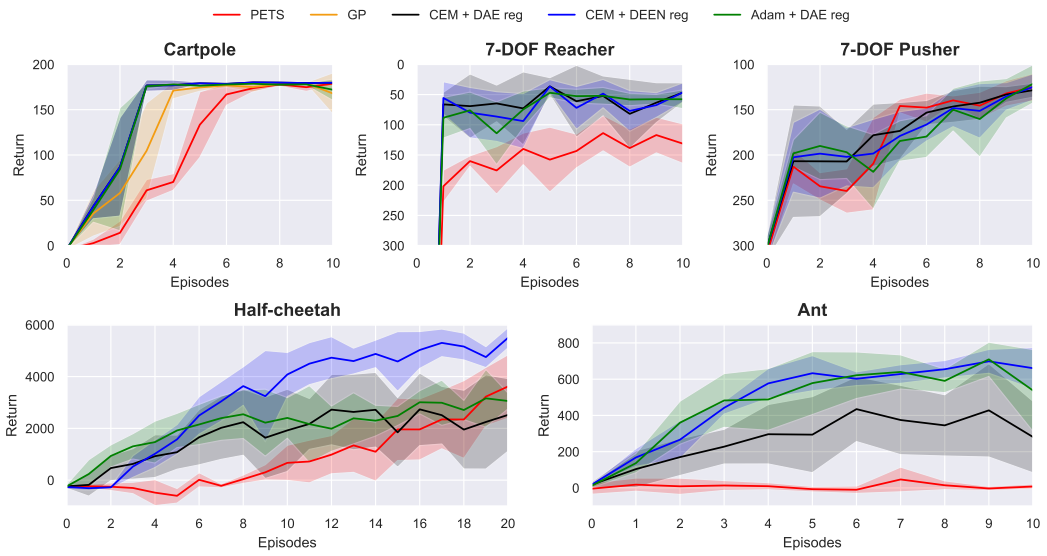

Figure 4: Results of our experiments on learning from scratch on five continuous control benchmarks: Cartpole, Reacher, Pusher, Half-cheetah and Ant. We compare to PETS [5] and DAE regularization [17] using the return (cumulative reward) obtained in each episode. PETS is a state-of-the-art model-based RL algorithm and DAE regularization has been shown to be effective in sample-efficient learning in these tasks. We also compare against planning with Gaussian Processes (GP) based dynamics model in the Cartpole task. We show the mean and standard deviation of each setting averaged across 5 seeds.

4.2 Experiments on Learning from Scratch

In this section, we test the effectiveness of regularization using energy-based models for learning from scratch. We perform one or more episodes of random exploration, train a dynamics model and the regularizer on the transitions and further interact with the environment using MPC by planning at each time-step using the regularized objective in Equation 4. We maintain a replay buffer of all past transitions and at the end of each episode the dynamics model and regularizer are re-trained on the whole replay buffer. In these experiments, we use a single feedforward network as the dynamics model (as opposed to an ensemble of models in [5]) to test the efficacy of the proposed method in a simple setting. The results are shown in Figure 4. In low-dimensional environments such as Cartpole, Reacher and Pusher, CEM optimization with DEEN regularization is comparable or better than the other methods. In Half-cheetah, CEM optimization with DEEN regularization clearly performs the best, obtaining good asymptotic performance in just 20,000 timesteps (corresponding to 16.6 minutes of experience). In Ant, CEM optimization with DEEN regularization also performs the best, learning to walk reasonably in just 4,000 timesteps (corresponding to 3.3 minutes of experience)[1]. It can also be observed that CEM optimization with DEEN regularization performs competitively or better than Adam optimization with DAE regularization, which requires much more computation. In the Half-cheetah benchmark, state-of-the-art model-free methods [23, 24] and PETS obtains better asymptotic performance during later stages of learning. We postulate that this is due to lack of a proper exploration mechanism in the proposed approach. However, DEEN regularization enables excellent performance during the early stages of training and also shows consistent improvements after each episode. This is very important for practical applications in the real-world, where the agent is expected to perform sensible actions from the very beginning. The proposed method enables efficient exploitation of the learned dynamics model and combining it with an explicit exploration mechanism would facilitate controlled exploration and exploitation. We leave complementation of the proposed method with an effective exploration strategy to future research. For example, energy estimates of the transitions could be used as bonuses for curiosity-driven exploration [25] to visit novel states and try novel actions.

[1] Videos of the training progress of the agents are included in the supplementary material.

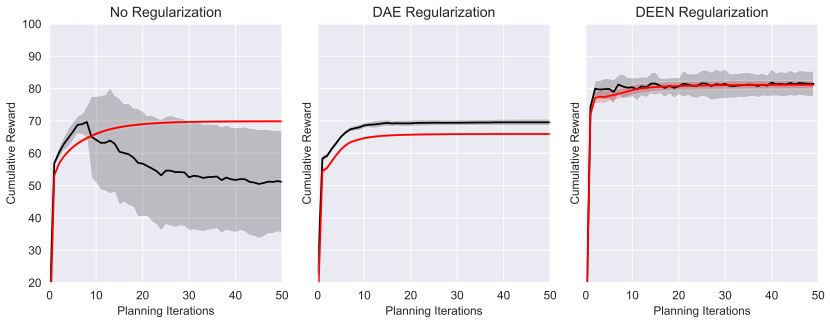

Figure 5: Visualization of trajectory optimization after 5 episodes of training in the Half-cheetah environment. We perform actions in the environment for 50 timesteps using MPC and then initialize the CEM optimization with the a random population of trajectories and optimize the planning objective in three different settings: 1) without any regularization, 2) with DAE regularization [17], and 3) with DEEN regularization. Here, the red lines denote the rewards predicted by the model (imagination) and the black lines denote the true rewards obtained when applying the sequence of optimized actions (reality). It is noticeable that planning without any regularization exploits inaccuracies of the dynamics model but DAE and DEEN regularization is able to prevent this.

We visually demonstrate the effectiveness of DEEN regularization in Figure 5. To compare with [17], we visualize trajectory optimization on the Half-cheetah task using dynamics models obtained after 5 episodes of training. We perform actions in the environment for 50 timesteps using model-based planning to arrive at a stable state and then perform trajectory optimization using a randomly initialized population of action trajectories. Without any regularization, the planning procedure leads to trajectories that are predicted to produce high rewards but is not the case in reality. It can be observed that while DAE regularization is also effective, DEEN regularization is clearly better, being able to successfully prevent the reality and imagination from diverging and also leading to trajectories with a better outcome.

4.3 Model Architecture and Hyperparameters

The important hyperparameters used in our experiments are reported in Table 2. Following [5, 17], we used a Bayesian neural network (BNN) with mean and variance predictions as our dynamics model. Although the predicted variance is not used for planning, it was found to have a regularizing effect during training [5]. We used a vanilla feedforward network to model the energy function.The energy estimation network is trained by corrupting the transitions in the replay buffer using additive Gaussian noise and minimizing the objective in Equation 7. We found the noise scale σ, cost multiplier α and number of training epochs to be the most sensitive hyperparameters. To prevent overfitting to the replay buffer, we explored simple strategies like increasing the number of initial episodes with random exploration and decaying the number of training epochs after each episode. In Half-cheetah, we perform random exploration for the first 3 episodes and decay the number of training epochs to 8 after 10 episodes. In Ant, we decay the number of training epochs of the dynamics model and the DEEN network by factors of 0.6 and 0.7 respectively.

5 Related Work

This paper is inspired by the recent works on planning with learned dynamics models. PILCO [26] is a well-known method for sample-efficient learning in low-dimensional control problems. However, it has been difficult to scale such methods to high-dimensional problems. Nagabandi et al. [8] used neural networks as dynamics models for planning to demonstrate sample-efficient learning on challenging continuous control problems and further fine-tuned the learned policy using model-free methods for good asymptotic performance. Mishra et al. [27] introduced deep generative models based on convolutional autoregressive models and variational autoencoders [28] and used it for gradient-based trajectory optimization and policy optimization. Chua et al. [5] used ensembling of probabilistic dynamics models and trajectory sampling to deal with inaccuracies of learned dynam-

Table 2: Important hyperparameters used in our experiments

	Hyperparameter	Cartpole	Reacher	Pusher	Half-cheetah	Ant
Model	Hidden layers	3	3	3	4	4
	Hidden size	200	200	200	200	400
	Epochs	500	500	100	300	600
	Batch Size	32	32	32	128	400
DEEN	Hidden layers	3	3	3	5	3
	Hidden size	200	200	200	500	300
	Epochs	500	500	100	100	800
	Batch Size	32	32	32	32	64
	Noise scale σ	0.1	0.1	0.1	0.37	0.9
	Cost multiplier α	0.001	0.001	0.01	0.05	0.035
CEM	Iterations	5	5	5	5	7

ics models and demonstrated sample-efficient learning to achieve high asymptotic performance on challenging continuous control problems. While the use of ensembling in [5] allows for a better dynamics model, energy-based regularization also bounds the transitions of the agent to be similar to it's previous experience, which might be important for safety-critical applications. While KL divergence penalty between action distribution has been used in previous works [29, 30], energy-based regularization also bounds the familiarity of states. Boney et al. [17] proposed to use denoising autoencoders to prevent trajectory optimization from exploiting modelling inaccuracies and demonstrated sample-efficient learning from very few episodes, using gradient-based trajectory optimization. Hafner et al. [31] introduced a novel latent dynamics model for planning in high-dimensional observation spaces, such as images. Recent works on model-based RL have also explored Dyna-style [32] architectures where learned dynamics models are used to generate data to train model-free methods [33, 34, 35].

Energy-based models can be directly used for planning by sampling future trajectories from the generative model. Du and Mordatch [36] showed that energy-based models can be used to sample diverse predictions of future trajectories and it was later extended in [37] for model-based planning. DEEN [12] could also be directly used for planning by sampling future trajectories using the novel walk-jump sampling algorithm introduced in [13]. However, sampling from such models at each timestep for planning is expensive and we instead use a separate forward dynamics model for directly predicting the future trajectories but only use the energy-based model for regularization. This can be seen as an ensemble of two different kinds of models.

6 Conclusion

Planning with learned dynamics models is challenging because planning can exploit the inaccuracies in the model to produce over-optimistic trajectories. In this paper, we propose to regularize planning using energy estimates of state transitions in the environment. We use a recently proposed energy estimation method called DEEN for this purpose. We demonstrated that an energy estimation network can be trained on the past experience of pre-trained dynamics models to further improve planning. We also demonstrated that the energy regularization enables sample-efficient learning on challenging tasks such as Half-cheetah and Ant, in just a few minutes of interaction.

One of the limitations of the proposed and related model-based planning algorithms is the additional hyperparameter tuning required for learning dynamics models. AutoML algorithms can be potentially used to automate the training of effective dynamics model by splitting the replay buffer into a training set and validation set and optimizing the prediction performance on the validation set. This could enable automatic architecture and hyperparameter tuning of dynamics models using more computational resources, without any additional data or human supervision. This would be an interesting line of future work.

Acknowledgments

We would like to thank Saeed Saremi for valuable discussions about his work on deep energy estimator networks and neural empirical Bayes.

References

[1] 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, 2015.

[2] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, et al. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354, 2017.

[3] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[4] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel. Benchmarking deep reinforcement learning for continuous control. In *International Conference on Machine Learning*, pages 1329–1338, 2016.

[5] 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.

[6] K. Lowrey, A. Rajeswaran, S. Kakade, E. Todorov, and I. Mordatch. Plan online, learn offline: Efficient learning and exploration via model-based control. In *International Conference on Learning Representations*, 2019.

[7] 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.

[8] 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.

[9] I. Clavera, A. Nagabandi, S. Liu, R. S. Fearing, P. Abbeel, S. Levine, and C. Finn. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. In *International Conference on Learning Representations*, 2019.

[10] C. E. Garcia, D. M. Prett, and M. Morari. Model predictive control: theory and practices - a survey. *Automatica*, 25(3):335–348, 1989.

[11] Y. LeCun, S. Chopra, and R. Hadsell. A tutorial on energy-based learning. 2006.

[12] S. Saremi, A. Mehrjou, B. Schölkopf, and A. Hyvärinen. Deep energy estimator networks. *arXiv preprint arXiv:1805.08306*, 2018.

[13] S. Saremi and A. Hyvärinen. Neural empirical Bayes. *arXiv preprint arXiv:1903.02334*, 2019.

[14] H. Robbins. An empirical Bayes approach to statistics. In *Proc. 3rd Berkeley Symp. Math. Statist. Probab., 1956*, volume 1, pages 157–163, 1956.

[15] K. Miyasawa. An empirical Bayes estimator of the mean of a normal population. *Bulletin of the International Statistical Institute*, 38(181-188):1–2, 1961.

[16] M. Raphan and E. P. Simoncelli. Least squares estimation without priors or supervision. *Neural computation*, 23(2):374–420, 2011.

[17] R. Boney, N. Di Palo, M. Berglund, A. Ilin, J. Kannala, A. Rasmus, and H. Valpola. Regularizing trajectory optimization with denoising autoencoders. In *Advances in Neural Information Processing Systems*, 2019.

[18] P. Parmas, C. E. Rasmussen, J. Peters, and K. Doya. PIPPS: Flexible model-based policy search robust to the curse of chaos. In *International Conference on Machine Learning*, pages 4062–4071, 2018.

[19] P. Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

[20] G. Alain and Y. Bengio. What regularized auto-encoders learn from the data-generating distribution. *The Journal of Machine Learning Research*, 15(1):3563–3593, 2014.

[21] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. OpenAI Gym. *arXiv preprint arXiv:1606.01540*, 2016.

[22] Z. I. Botev, D. P. Kroese, R. Y. Rubinstein, and P. L'Ecuyer. The cross-entropy method for optimization. In *Handbook of statistics*, volume 31, pages 35–59. Elsevier, 2013.

[23] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1856–1865, 2018.

[24] S. Fujimoto, H. Hoof, and D. Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pages 1582–1591, 2018.

[25] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 16–17, 2017.

[26] 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.

[27] N. Mishra, P. Abbeel, and I. Mordatch. Prediction and control with temporal segment models. In *International Conference on Machine Learning*, pages 2459–2468, 2017.

[28] D. P. Kingma and M. Welling. Auto-encoding Variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[29] S. Levine and V. Koltun. Guided policy search. In *International Conference on Machine Learning*, pages 1–9, 2013.

[30] 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.

[31] 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*.

[32] 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.

[33] T. Kurutach, I. Clavera, Y. Duan, A. Tamar, and P. Abbeel. Model-ensemble trust-region policy optimization. In *International Conference on Learning Representations*, 2018.

[34] I. Clavera, J. Rothfuss, J. Schulman, Y. Fujita, T. Asfour, and P. Abbeel. Model-based reinforcement learning via meta-policy optimization. In *Conference on Robot Learning*, pages 617–629, 2018.

[35] D. Ha and J. Schmidhuber. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems*, pages 2450–2462, 2018.

[36] Y. Du and I. Mordatch. Implicit generation and generalization in energy-based models. *arXiv preprint arXiv:1903.08689*, 2019.

[37] Y. Du, T. Lin, and I. Mordatch. Model based planning with energy based models. *ICML 2019 Workshop on Generative Modeling and Model-Based Reasoning for Robotics and AI*, 2019.