Imagination-Based Decision Making with Physical Models in Deep Neural Networks

Jessica B. Hamrick
University of California, Berkeley & Google DeepMind
jhamrick@berkeley.edu

Razvan Pascanu, Oriol Vinyals, Andy Ballard, Nicolas Heess, Peter Battaglia
Google DeepMind
{razp,vinyals,aybd,heess,peterbattaglia}@google.com

Abstract

Decision-making is challenging in continuous settings where complex sequences of events determine rewards, even when these event sequences are largely observable. In particular, traditional trial-and-error learning strategies may have a hard time associating continuous actions with their reward because of the size of the state space and the complexity of the reward function. Given a model of the world, a different strategy is to use *imagination* to exploit the knowledge embedded in that model. In this regime, the system directly optimizes the decision for each episode based on predictions from the model. We extend deep learning methods that have been previously used for model-free learning and apply them towards a model-based approach in which an *expert* is consulted multiple times in the agents' imagination before it takes an action in the world. We show preliminary results on a difficult physical reasoning task where our model-based approach outperforms a model-free baseline, even when using an inaccurate expert.

1 Introduction

While significant advances in deep learning have been made in areas of reinforcement learning [1, 2] and control [3], most efforts focus on optimization during learning rather than at decision time. Online optimization methods, in which an agent can compute the best course of action on-the-fly, have been explored extensively in traditional machine learning, but have received little attention by deep learning-based efforts. Only recently has any work begun to address the problem of online computation in deep neural networks at all. For example, [4] proposed a method for *adaptive computation time* (ACT) in which the network learns to spend more time on more difficult problems. However, this approach assumes that the result of the computation will be an expectation, rather than an optimum. [5] trained a network to perform gradient descent updates for another network, outperforming existing black-box optimizers on several regression and classification tasks. To our knowledge, however, no one has yet investigated methods for online optimization in a deep learning regime during planning- or control-based tasks.

We present a method for model-based decision making in neural networks in which the optimization occurs online, and evaluate our approach on a difficult physical reasoning task. In this task, a force needs to be applied to a spaceship such that it will arrive at a particular location in space after a certain amount of time (Figure 1a). Importantly, the gravity of the surrounding planets affects the trajectory of the spaceship in highly nonlinear ways. We show that a model-free parameterized controller performs poorly on this task, but when it is allowed to perform additional online computation—i.e., trying out multiple actions using a model, which we term an *expert*—its performance improves significantly, even when the expert is inaccurate.

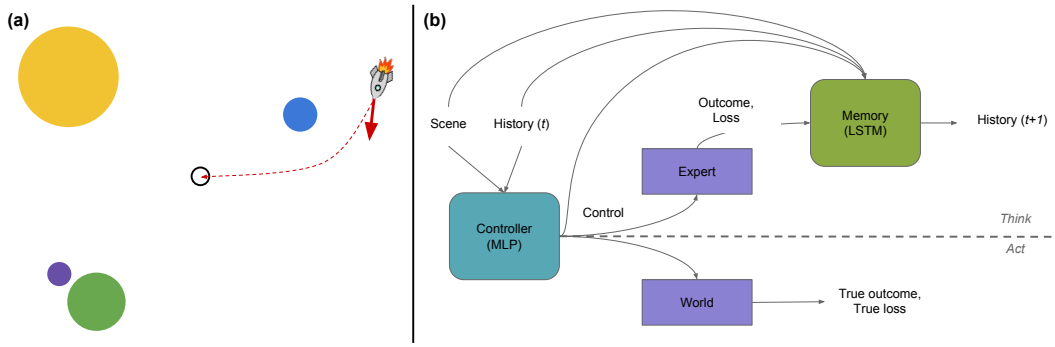

Figure 1: **Spaceship task and model architecture.** Left: Scenes consisted of a number of planets (depicted here by colored circles) of different masses, as well as a spaceship (also with a variable mass). The task was to apply a force to the spaceship for one time step of simulation (depicted here as a solid red arrow) such that the resulting trajectory (dotted red arrow) would put the spaceship at a target (black circle) after 11 steps of simulation. Right: The architecture of our iterative controller, as described in Section 2.2.

2 Imagination-Based Decision Making

2.1 Theory

We consider a class of real-valued decision making tasks in which the goal is to find a *control* that minimizes some distance to a target state x^* given an initial state x. The state resulting from the control is given by a forward process f, i.e. $x' = f(x, c)$. The goal is to find the optimal control $c^* = \arg\min_c \mathcal{L}(x^*, f(x, c))$, where $\mathcal{L}$ is some loss function with respect to a target state x^*.

As a baseline, we consider a traditional model-free approach in which we find a direct mapping from x^* and x to controls. We refer to this solution as a *reactive* controller $c = \mathcal{C}(x^*, x)$, such that the loss is minimized by c. This approach, while simple, may not scale well: as the complexity of the state space increases, a reactive controller with a fixed number of parameters will do an increasingly poor job because the complexity requires more expressivity, which requires more parameters. While this decrease in performance can be mitigated by increasing the network size and hand-tuning the architecture, it can also mean a substantial increase in the amount of data needed for training.

To improve upon the reactive controller, we introduce an iterative method that optimizes over a model of f. We term such a model an *expert*, denoted by $\mathcal{E}$. An *iterative* controller $\mathcal{C}$ can take a *history* of controls and results, $H_{n-1} = \{(c_0, \mathcal{E}(x, c_0)), \ldots, (c_{n-1}, \mathcal{E}(x, c_{n-1})\}$, and based on these examples, suggest a new control, $c_n = \mathcal{C}(x^*, x, H_{n-1})$. Here, n is the number of steps of iteration, which we refer to as *ponder* steps (after [4]). Under this framework, the reactive controller can be defined as an iterative controller with $H = \emptyset$. Thus, this approach subsumes traditional model-free approaches, combining a model-free policy for control with a policy for model-based optimization.

2.2 Implementation

Our implementation of the iterative controller is depicted in Figure 1b and can be summarized as:

$$c_n = \mathcal{C}(x^*, x, H_{n-1}; \theta_\mathcal{C}) \qquad \widetilde{H}_n = \mathcal{M}(x, \widetilde{H}_{n-1}, c_n, \mathcal{E}(x, c_n), \mathcal{L}(x^*, \mathcal{E}(x, c_n)); \theta_\mathcal{M}) \qquad (1)$$

where $\mathcal{M}$ is the memory, $\mathcal{E}$ is the expert, $\theta_\mathcal{C}$ are the parameters of the controller, $\theta_\mathcal{M}$ are the parameters of the memory, and $\widetilde{H}_n$ is an approximation of H_n with fixed dimensionality. The final control c_n is executed in the world, and the experts are trained on the resulting outcome by minimizing $\mathcal{L}_\mathcal{E}(f(x, c_n), \mathcal{E}(x, c_n; \theta_\mathcal{E}))$, where $\mathcal{L}_\mathcal{E}$ is loss function for the expert (e.g., distance between the predicted location of the spaceship and the true location), and $\theta_\mathcal{E}$ are the parameters of the expert. The loss for the memory and controller is $\mathcal{L}(x^*, f(x, c_n))$, as previously described. We note that this assumes that we can obtain gradients from f, which is a strong assumption and not typically true. We make the assumption here in order to demonstrate that our approach works in the best case. However, if this assumption is not valid, then the network can still be trained using policy gradient algorithms, which are known to work well in practice [6, 7].

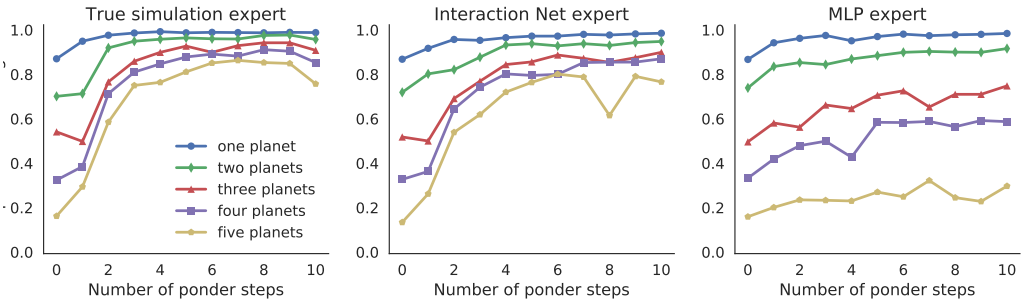

Figure 2: **Test performance of the reactive and iterative controllers.** Each line corresponds to the performance of an iterative controller trained for a fixed number of ponder steps on one of the five datasets; the line color indicates which dataset the controller was trained on. Performance is defined as the proportion of scenes in which the spaceship came within a small radius of the target. Left: performance using the true simulation expert. Middle: performance using the interaction net expert. Right: performance using the MLP expert.

The controller is a two-layer perceptron (MLP), each with 100 units: the first layer is a standard layer with a ReLU activation function, while the second layer has a multiplicative interaction similar to [8], which we found to work better in practice. The memory, which compresses the history to a vector of fixed dimensionality, is a single Long Short-Term Memory (LSTM) [9] layer of size 100.

We implemented three different experts. The *true simulation* expert is the same as the world model, and consists of a simulation for 11 timesteps. The *interaction net* expert is an interaction network [10], which has previously been shown to be able to learn to predict n-body dynamics accurately for simple systems. The interaction network consists of a relational module and an object module. In our case, the relational module was composed of 4 hidden layers of 150 nodes each, outputting "effects" encodings of size 100. These effects, together with the relational model input are then used as input to the object model, which contained a single hidden layer of 100 nodes. The object model outputs the velocity of the spaceship and we trained it to predict the velocity on every timestep of the spaceship's trajectory. The *MLP* expert is a two-layer MLP (with the same architecture as the controller) that is trained to predict only the final location of the spaceship.

3 Experiments

3.1 Training

We generated five datasets each of 100,000 training scenes and 1,000 testing scenes. Each dataset contained scenes with a different number of planets, ranging from a single planet to five planets. The planets were always fixed (i.e., they could not move), and the spaceship always started at the beginning of each episode with zero velocity. We simulated our scenes using a physical simulation of gravitational dynamics with the Euler method of integration, implemented in TensorFlow [11].

The iterative controller was trained to take a fixed number of ponder steps, ranging from 0 (i.e., the reactive controller) to 10. The interaction network and MLP experts were trained simultaneously with the rest of the network, but using a different step size than the controller and memory. We trained for 100,000 iterations over minibatches of size 1000.

3.2 Results

Figure 2 shows the performance on the test set of the reactive and iterative controllers for different numbers of ponder steps. The reactive controller fares poorly on the task, especially as the task becomes more difficult: with the five planets dataset, it is only able to achieve a success rate of 15.9% on average. In contrast, the iterative controller with the true simulation expert is able to perform significantly better, reaching ceiling performance on the one and two planets datasets, and a peak 87.0% success rate on the five planets dataset.

Figure 2 also demonstrates that the choice of expert is very important. With the interaction net expert, the iterative controller achieves a maximum success rate of 80.8% on the five planets dataset. This high level of performance is possible because the interaction net closely mirrors the dynamics of the true simulation expert. With the MLP expert, however, performance is significantly diminished, only reaching a maximum of 32.9% on the five planets dataset. Despite the weaker performance of this inaccurate expert, we emphasize that there is still a significant benefit to be gained by pondering: with even a single ponder step, the MLP expert is able to outperform the reactive controller. These results indicate that pondering—even with a highly inaccurate model—can still lead to better performance than a model-free approach.

4 Discussion

We have presented a new, deep learning approach to decision-making using online optimization with a physical model. This idea of "imagination-based decision making" not only outperforms our model-free baseline, but also agrees more closely with how humans reason about complex physical systems [12]. Importantly, by imagining what will happen, our approach allows agents to test out actions to evaluate their consequences before actually executing them. Additionally, our approach suggests a way to bridge between discriminative deep learning models for physical reasoning (e.g., [13]) and generative simulation-based models (e.g., [12]). By embedding powerful, deep generative models such as interaction nets [10] in decision-making frameworks like the one presented here, we can exploit the power of the deep learning approach without sacrificing the flexibility of simulation.

Acknowledgments

We thank Andrea Tacchetti, Tom Erez, Nando de Freitas, Guillaume Desjardins, Joseph Modayil, Hubert Soyer, Alex Graves, David Reichert, Theo Weber, Jon Scholz, Will Dabney, and many others on the DeepMind team for helpful discussions and feedback.

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*, vol. 518, no. 7540, pp. 529–533, 2015.

[2] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[3] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *Journal of Machine Learning Research*, vol. 17, pp. 1–40, 2016.

[4] A. Graves, "Adaptive computation time for recurrent neural networks," *arXiv:1603.08983*, 2016.

[5] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, and N. de Freitas, "Learning to learn by gradient descent by gradient descent," *arXiv:1606.04474*, 2016.

[6] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3-4, pp. 229–256, 1992.

[7] 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:1509.02971*, 2015.

[8] A. van den Oord, N. Kalchbrenner, O. Vinyals, L. Espeholt, A. Graves, and K. Kavukcuoglu, "Conditional Image Generation with PixelCNN Decoders," *arXiv:1606.05328*, 2016.

[9] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[10] P. Battaglia, R. Pascanu, M. Lai, D. J. Rezende, and K. Kavukcuoglu, "Interaction networks for learning about objects, relations and physics," *Advances in Neural Information Processing Systems*, 2016.

[11] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. Software available from tensorflow.org.

[12] P. W. Battaglia, J. B. Hamrick, and J. B. Tenenbaum, "Simulation as an engine of physical scene understanding," *Proceedings of the National Academy of Sciences*, vol. 110, no. 45, pp. 18327–18332, 2013.

[13] A. Lerer, S. Gross, and R. Fergus, "Learning Physical Intuition of Block Towers by Example," *arXiv:1603.01312*, 2016.