Proceedings of Machine Learning Research vol 120:1–10, 2020 2nd Annual Conference on Learning for Dynamics and Control

Structured Mechanical Models for Robot Learning and Control

Jayesh K. Gupta* JKG@CS.STANFORD.EDU
Kunal Menda* KMENDA@STANFORD.EDU
Zachary Manchester ZACMANCHESTER@STANFORD.EDU
Mykel J. Kochenderfer MYKEL@STANFORD.EDU
Stanford University

Editors: A. Bayen, A. Jadbabaie, G. J. Pappas, P. Parrilo, B. Recht, C. Tomlin, M. Zeilinger

Abstract

Model-based methods are the dominant paradigm for controlling robotic systems, though their efficacy depends heavily on the accuracy of the model used. Deep neural networks have been used to learn models of robot dynamics from data, but they suffer from data-inefficiency and the difficulty to incorporate prior knowledge. We introduce Structured Mechanical Models, a flexible model class for mechanical systems that are data-efficient, easily amenable to prior knowledge, and easily usable with model-based control techniques. The goal of this work is to demonstrate the benefits of using Structured Mechanical Models in lieu of black-box neural networks when modeling robot dynamics. We demonstrate that they generalize better from limited data and yield more reliable model-based controllers on a variety of simulated robotic domains.

Keywords: System Identification; Model-based Control; Physics-based Model Learning; Deep Neural Networks

1. Introduction

Model-based optimal control has been extensively used in robotics. However, the efficacy of this approach can depend heavily on the quality of the model used to predict how control inputs affect the dynamics of the robot. While system identification methods can be used to improve the accuracy of a model, models derived from first-principles often fail to accurately express complex phenomena such as friction or aerodynamic drag, suffering from model *bias*. Although there have been proposals to learn *offset functions* that correct for such model bias, they are usually restricted to specific use cases (Ratliff et al., 2016; Nguyen-Tuong and Peters, 2010). To avoid such bias, black-box models such as deep neural networks have been proposed as representations for robot dynamics, with their parameters trained using data collected from the robotic system (Chua et al., 2018; Lambert et al., 2019).

Black-box neural networks (BBNNs) typically take as input the robotic system's state and control input, and predict its state at the next time-step. Though expressive enough to model arbitrary phenomena, such a parameterization has shortcomings. BBNNs require large amounts of training data to generalize well to unseen states, and such data can be

* Equal contribution

expensive to acquire on real robotic systems. The data-inefficiency of these models is a result of the fact that they do not easily incorporate prior knowledge, and are not biased toward solutions corresponding to physical systems. Furthermore, many robotic control frameworks, such as feedback linearization, leverage structure present in models derived from first principles (Siciliano and Khatib, 2008) and cannot be easily used with BBNNs.

In this work, we present Structured Mechanical Models (SMMs), which are a flexible black-box parameterization of mechanical systems. Derived from Lagrangian mechanics, SMMs explicitly encode the structure present in the manipulator equation (Murray et al., 1994). SMMs use the Deep Lagrangian Network architecture (Lutter et al., 2019b) to predict the system's Lagrangian, and use standard neural networks to predict input forces and dissipative forces such as friction. By parameterizing these different components separately, SMMs retain the expressivity of BBNNs but encode a physical prior that improves their data-efficiency. Furthermore, SMMs can be used with a wide variety of model-based control frameworks, including feedback-linearization and energy-based control (Lutter et al., 2019a).

The goal of the paper is to empirically demonstrate that SMMs are a better black-box parameterization for robotic systems than BBNNs. Given finite training data, we define "better" in the following ways:

- SMMs have lower generalization error on unseen states,
- SMMs yield more reliable model-based controllers,
- SMMs are able to incorporate prior knowledge if available,
- SMMs are amenable to the full suite of model-based control techniques.

We empirically validate the first two claims on a suite of simulated robotic domains, including a Furuta Pendulum, Cartpole, Acrobot, and Double Cartpole. We then discuss the remainder of the claims, potential limitations of SMMs, and how those limitations can be overcome.

2. Black-box parameterizations of robotic systems

In this section, we discuss different methods for parameterizing the dynamics of a robotic system without assuming any prior knowledge.

2.1. Dynamical and Mechanical Systems

Dynamical systems are systems with state $x \in \mathbb{R}^n$ that evolve over time. The time-rate-of-change of the state $\dot{x}$ is specified by a function $\dot{x} = f(x, u)$, and $u \in \mathbb{R}^m$ is a control input. Mechanical systems are a subset of dynamical systems describing the evolution of extended bodies in the physical world, and can describe many robotic systems of interest. The state x of a mechanical system is composed of a set of *generalized coordinates* $q \in \mathbb{R}^{n_q}$ and their corresponding time-rates-of-change $\dot{q} \in \mathbb{R}^{n_q}$, called the *generalized velocities*. Akin to Newton's second law (force equals mass times acceleration), the dynamics of a mechanical system prescribe the *generalized acceleration*, i.e. $\ddot{q}$, as follows (Murray et al., 1994):

$$\mathbf{M}(q)\ddot{q} = F(q, \dot{q}, u) - \mathbf{C}(q, \dot{q})\dot{q} - G(q) \tag{1}$$

Here, $\mathbf{M}(q)$ is the body's *mass matrix*. The right-hand side of the equation is composed of the forces that act on the system. The term $\mathbf{C}(q, \dot{q})\dot{q}$ is the *Coriolis force*, and is entirely a function of $\mathbf{M}(q), q$ and $\dot{q}$. The terms $G(q)$ and $F(q, \dot{q}, u)$ represent the *conservative* and *dissipative* forces, respectively. The conservative forces $G(q) = \nabla_q V(q)$ are equal to the gradient of the potential energy $V(q)$, and are termed *conservative* forces because they do not change the total energy of the system. The dissipative forces are forces such as friction and torques from actuators, which do change the total energy of the system. For most real-world mechanical systems, we can write the dissipative forces in *control-affine* form, i.e.:

$$F(q, \dot{q}, u) = \tilde{F}(q, \dot{q}) + \mathbf{B}(q) \cdot u \tag{2}$$

Here, the control input u is mapped to a force through the *input Jacobian* $\mathbf{B}(q)$, and all other dissipative forces are captured by $\tilde{F}(q, \dot{q})$. Control-affine formulations are convenient for use with a class of control techniques called *feedback linearization*, where $B(q)$ is inverted to yield a linear control law.

Left-multiplying Equation (1) by $\mathbf{M}^{-1}(q)$, we can isolate $\ddot{q}$ and define $\dot{x} = [\dot{q}, \ddot{q}]$ in terms of $\mathbf{M}(q)$, $V(q)$, and $F(q, \dot{q}, u)$. This system is defined using a *continuous-time* (CT) formulation. We can convert a CT dynamical system into a *discrete-time* (DT) dynamical system ($x_{t+\Delta t} = f^d(x_t, u_t)$) using an integration scheme such as Runge-Kutta (Runge, 1895).

2.2. Black-box Models

The dynamics of a mechanical system are often derived from first-principles (Murray et al., 1994; Nguyen-Tuong and Peters, 2011), but such models can often fail to capture phenomena present in the real world, limiting the efficacy of model-based controllers that use them.

Deep neural networks are a flexible function class that can represent models of almost arbitrary phenomena given training data, and have been used extensively in the field of computer vision (Krizhevsky et al., 2012). Neural networks have been used to represent dynamical systems (Funahashi and Nakamura, 1993; Kuschewski et al., 1993; Morton et al., 2018; Chua et al., 2018; Nagabandi et al., 2019), and these formulations have been used to perform model-based control (Lambert et al., 2019). In these formulations, a neural network directly maps the inputs x_t and u_t to the state derivative $\dot{x}_t$ or next state $x_{t+\Delta t}$. We call this representation a Black-Box Neural Network (BBNN). Though flexible enough to represent an arbitrary dynamical system, such a representation does not encode the structure shared by dynamics of all mechanical systems, described in Section 2.1. As a result, BBNNs typically require more data to train, and cannot leverage any prior knowledge one might have about the structure of the mechanical system. Furthermore, since such models lack the explicit structure of mechanical systems, they cannot be used with control techniques such as feedback linearization (Siciliano and Khatib, 2008) or energy-based control (Spong, 1996).

2.3. Deep Lagrangian Networks

To recover the structure present in the manipulator equation, Deep Lagrangian Networks use neural networks to parameterize the mass-matrix $\mathbf{M}_\theta(q)$ and potential energy of $V_\theta(q)$ of the robotic system, where θ represents the neural network parameters (Lutter et al., 2019b).

The Coriolis forces can be computed using $\dot{q}$ and the Jacobian of the mass-matrix with respect to q (Murray et al., 1994; Lutter et al., 2019b). Such model parameterizations have been used to model real robot arms (Lutter et al., 2019b) and for energy-based control of real Furuta pendulums (Lutter et al., 2019a). In these works, dissipative forces are assumed to be directly measured, and not predicted. In the next section, we present Structured Mechanical Models (SMMs), which use neural networks to predict dissipative forces as well as Deep Lagrangian Networks, as a general framework for modeling mechanical systems.

3. Structured Mechanical Models

SMMs are models of mechanical systems that separately parameterize the system's mass-matrix, potential energy, and dissipative and input forces. If we have no prior knowledge of the system, we use Deep Lagrangrian Networks to parameterize the mass-matrix and potential energy of the system, as well as a neural network $F_\theta(q, \dot{q}, u)$ that predicts the combination of dissipative and input forces. This parameterization is as expressive as a BBNN that predicts $\ddot{q}$ as a function of q and $\dot{q}$ and u. This is because if we learn $M_\theta(q) = \mathbf{I}$ and $V_\theta(q) = 0$, then the SMM predicts $\ddot{q} = F_\theta(q, \dot{q}, u)$, which is equivalent to a BBNN.

However, almost all mechanical systems are control-affine, i.e. control inputs are mapped to forces via a Jacobian that only depends on q. Thus, a better black-box parameterization for robotic systems decomposes $F_\theta(q, \dot{q}, u)$ into $\mathbf{B}_\theta(q) \cdot u$ and $\tilde{F}_\theta(q, \dot{q})$ according to Equation (2), where $\mathbf{B}_\theta(q)$ is a neural network predicting the control-input Jacobian and $\tilde{F}_\theta(q, \dot{q})$ is a prediction of dissipative forces not resulting from the input. We call this control-affine parameterization SMM-C, and Figure 1 summarizes the networks used in both the BBNN and SMM-C parameterizations.

3.1. Parameter Optimization

Given a dataset $\mathcal{D} = \{\ldots, (x_t, u_t, x_{t+\Delta t}), \ldots\}$ of states, inputs, and next states, we find the maximum-likelihood (ML) parameters of an SMM or BBNN by minimizing the mean-squared prediction error of the next state. That is, we find:

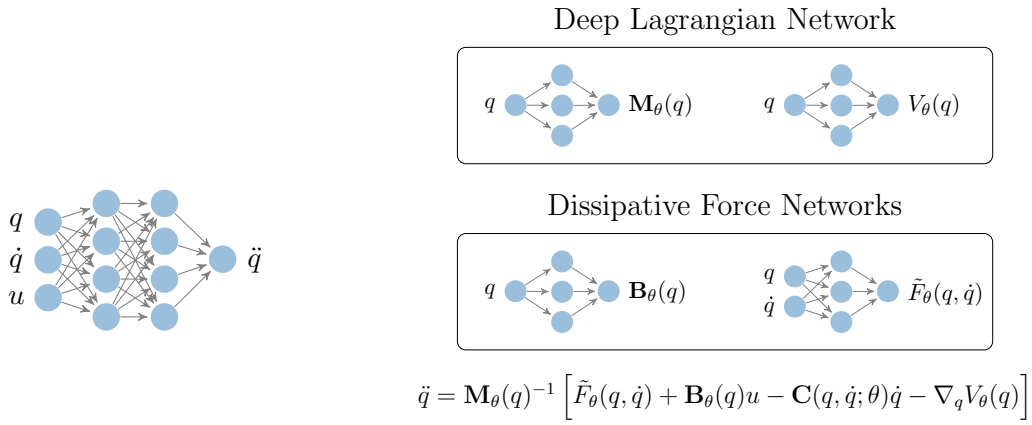
$$\theta^{\text{ML}} = \arg\min_\theta \quad \mathbf{E}_\mathcal{D}\left[x_{t+\Delta t} - f_\theta^d(x_t, u_t)\right] \tag{3}$$

As previously stated, $f_\theta^d(x_t, u_t)$ is a DT model of a dynamical system. If the model is a CT model, as is the case for SMMs and can be the case for BBNNs, then we apply an integration scheme to convert it to discrete-time.

4. Experiments

This section justifies the following claims. Given a finite amount of data gathered from a robotic system:

- SMMs typically have lower generalization error on unseen states than BBNNs, and,
- SMMs generally yield more reliable model-based controllers than BBNNs.



(a) BBNN (b) SMM-C

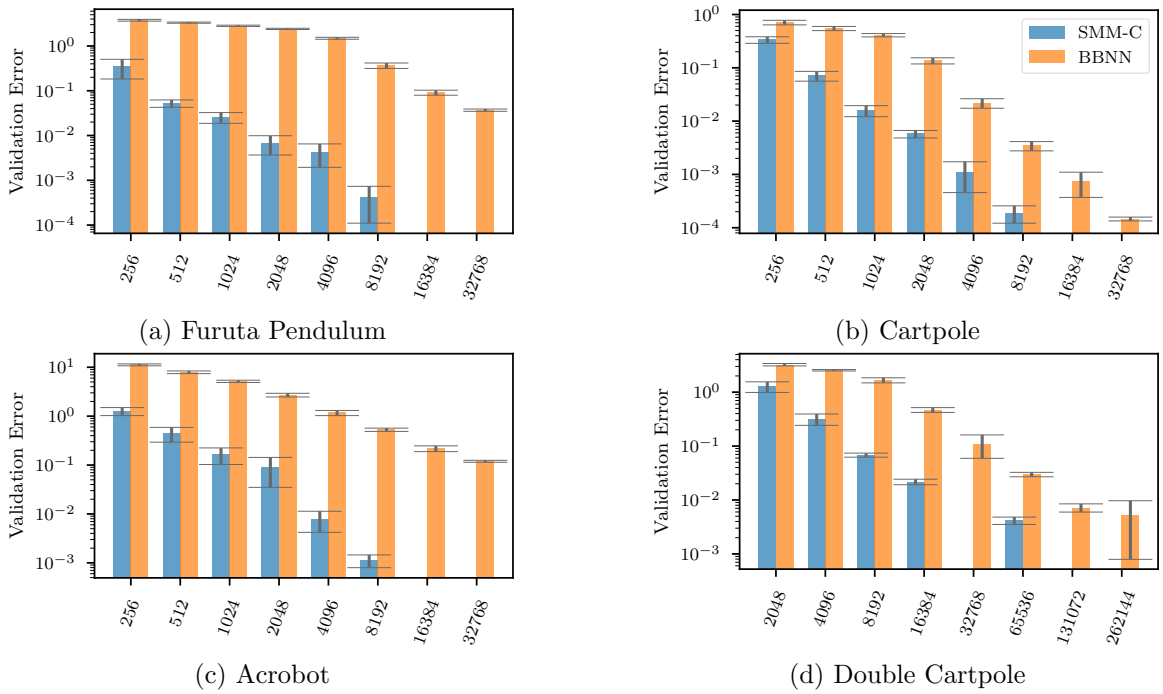
Figure 1: (a) A black-box model of dynamical system and (b) a Structured Mechanical Model of a control-affine system.

We demonstrate these claims on four simulated domains—Furuta Pendulum, Acrobot, Cartpole, and Double Cartpole (i.e. Cartpole with two poles attached in series). In both experiments, we only make the assumption that the robotic system is control-affine using the SMM-C model parameterization.

4.1. Generalization Error on Unseen States

In order to test whether SMM-Cs have better generalization error on unseen states, we uniformly sample state, input, and next-state tuples to form training and test datasets. We then train both an SMM-C model and a BBNN model on the dataset, and measure its generalization error as the mean-squared prediction error on the test set. We use training sets ranging from 256 to 32768 tuples and a test set with 32768 tuples. We run this experiment with 5 random seeds and present standard deviations for performance estimates.

Results: Figure 2 summarizes the generalization error of a BBNN and SMM-C parameterization of the dynamics of the four robotic domains, for various dataset sizes. We see that for all domains and training set sizes, the generalization error of the SMM-C parameterization is at least an order of magnitude better, and up to three orders of magnitude better than that of the BBNN parameterization. The disparity between the models' abilities to generalize from limited data can be explained by the following two facts: Firstly, BBNNs confound the forces generated by Coriolis effects, potential fields, control inputs, and dissipative sources, unlike SMMs which explicitly decouple these effects, and therefore serve as a better prior for the true model structure. Secondly, neural networks in SMM-Cs do not have any explicit dependence on the generalized velocities, aside from the function $\tilde{F}(q, \dot{q})$ that predicts disspative forces. As a result, SMM-Cs generalize far better to states with unseen velocities. Next, we show that even if enough data is provided such that an SMM-C and BBNN achieve similar generalization errors, SMM-Cs are more reliable when used for model-based control.



(a) Furuta Pendulum (b) Cartpole

(c) Acrobot (d) Double Cartpole

Figure 2: Generalization error of SMM-Cs and BBNNs for various training dataset sizes on four robotic domains.

4.2. Reliability of Model-Based Control

In this experiment, we attempt to solve Furuta Pendulum, Cartpole and Acrobot swing-up tasks by using the SMM-C and BBNN models trained in Section 4.1. That is, starting at their respective stable equilibria, a control sequence must be applied to bring the systems to and stabilize them at their respective unstable equilibria. These tasks are canonical examples of model-based control problems because they are underactuated, i.e. the control input cannot independently actuate the degrees-of-freedom of the robot. Though many approaches to model-based control exist, a popular approach to solving underactuated tasks is to first compute a *nominal trajectory* and then follow the nominal trajectory.

The nominal trajectory $(\bar{x}_{1:T}, \bar{u}_{1:T-1})$ is a series of states and inputs that are both dynamically-feasible according to the model, i.e. $x_{t+1} = f_\theta^d(x_t, u_t)$, and connect the initial and goal configurations with minimum cost. It is common to minimize a quadratic cost of the form:

$$J(x_{1:T}, u_{1:T-1}) = (x_T - x_g)^\top Q_F (x_T - x_g) + \sum_{t=1}^{T-1} (x_t - x_g)^\top Q (x_t - x_g) + u_t^\top R u_t \quad (4)$$

Here, x_g corresponds to the goal-state, and Q, Q_F, and R are positive definite matrices trading-off the trajectory's efficiency and control effort. In our experiments, we use Direct-Collocation Trajectory Optimization (DIRCOL) (Kelly, 2017; Howell et al., 2019) to find the nominal trajectory by solving a constrained non-linear program.

	SMM-C	BBNN	BBNN-32768	True
Furuta Pendulum	$(5.24 \pm 3.14) \times 10^{-3}$	$(6.29 \pm 8.98) \times 10^{0}$	*	$(7.86 \pm 0.0) \times 10^{-3}$
Cartpole	$(1.35 \pm 0.99) \times 10^{-3}$	$(1.89 \pm 1.14) \times 10^{1}$	$(1.80 \pm 1.17) \times 10^{1}$	$(6.40 \pm 0.0) \times 10^{-4}$
Acrobot	$(1.55 \pm 2.94) \times 10^{-1}$	$(0.84 \pm 1.03) \times 10^{1}$	*	$(1.48 \pm 0.0) \times 10^{-4}$

Table 1: Performance of learned SMM-C and BBNN models on the Furuta Pendulum, Cartpole and Acrobot swing-up tasks, compared to the true model.

Once the nominal trajectory is found, it must be followed by the robotic system in a manner that rejects disturbance. To do so, a feedback controller of the form $u_t = \pi(x_t, t) = \bar{u}_t + K_t(\bar{x}_t - x_t)$ is used to compute that input u_t at time t. The *feedback gains* K_t are computed using a Time-Varying Linear Quadratic Regulator (TVLQR) (Anderson, 2007). The resulting controller is then run on the true system, and the quality of the resulting trajectory is largely dependent on the accuracy of the model used in DIRCOL and TVLQR. Letting the resulting trajectory be $(x^{\pi}_{1:T}, u^{\pi}_{1:T-1})$, we can measure the *cost* of the trajectory as:

$$J^{\pi}(x^{\pi}_{1:T}, u^{\pi}_{1:T-1}) = \min_{x^{\pi}_t} \quad \|x^{\pi}_t - x_g\|_2^2 \tag{5}$$

If a model class is able to consistently generate policies that solve the swing-up task, the resulting cost will be low, and we will consider the model class reliable.

We note that TVLQR synthesizes the gains K_t using cost matrices $Q^{\text{TV}}, R^{\text{TV}}$, and Q_F^{TV} and the resulting cost J^{π} is sensitive to the choice of these hyperparameters. To make a fair comparison between model classes, we perform a grid search over these hyperparameters and report the lowest cost found.

In this experiment, we compare the five SMM-C and BBNN models trained with 8192 samples on each domain. In addition, we will also compare the five BBNN models trained on Cartpole with 32768 samples, referred to as BBNN-32768, since it has a generalization error comparable to that of the SMM-C models we are using. By doing so, we hope to explore if the structure present in SMMs aids model-based control beyond merely having low prediction error on unseen states. We report the mean and standard deviation of J^{π} across the five models.

Results: Table 1 summarizes the performance of model-based controllers synthesized using BBNNs and SMM-Cs on the two robotic domains tested. We see that SMM-Cs yield model-based controllers that achieve far lower costs when attempting to solve swing-up tasks, and is comparable in cost to the true model[1]. This observation can largely be explained by the fact that the BBNN models tested have worse generalization error than the SMM-C models tested.

However, we see that even the BBNN-32768 models, which have marginally better generalization errors than the SMM-C models trained on Cartpole ($(1.2 \pm 0.1) \times 10^{-4}$ for BBNN-32768 and $(1.9 \pm 0.7) \times 10^{-4}$ for SMM-C), perform worse than the SMM-C models. We hypothesize that in order to perform DIRCOL and TVLQR, we not only need accurate predictions of next states given unseen states and inputs, but also require accurate estimates of Jacobians with respect to the states and inputs. We find that on

1. Videos of the attempted swing-ups and source code for the experiments can be found at https://sites.google.com/stanford.edu/smm/.

the same test-set $\mathcal{D}_{\text{test}}$ used in Section 4.1, the Frobenius-norm error in predictions of $\nabla_x f_\theta^d(x,u)$ is $(4.1 \pm 6.7) \times 10^{-2}$ and of $\nabla_u f_\theta^d(x,u)$ is $(1.2 \pm 1.7) \times 10^{-4}$ for the SMM-C, but is $(0.5 \pm 1.1) \times 10^{-1}$ and $(0.5 \pm 2.9) \times 10^{-2}$ respectively for the BBNN-32768 trained on Cartpole, which is consistent with our hypothesis. Hence, we see evidence that SMM-Cs are a better black-box parameterization for model-based controllers.

5. Discussion and Conclusion

In the last section, we empirically validated the first two claims—that SMM-Cs are a better black-box parameterization than BBNNs in terms of generalization to unseen states, and yield more reliable model-based controllers. Though we used black-box parameterizations that make no assumptions about the system aside from that it is control-affine, SMM-Cs have the additional benefit of being able to incorporate any prior knowledge available to a practitioner. For example, we may know the system's mass matrix or kinematic structure and can replace the neural network predictor with a model derived from first principles. Additionally, we may want to model specific dissipative forces, like linear joint friction, instead of a black-box predictor, and so we can replace $\tilde{F}_\theta(q, \dot{q})$ with a more structured parameterization. Lastly, SMM-Cs can be used with the full suite of optimal controllers. Since the model is a control-affine system, they can easily be used with feedback-linearization techniques, unlike BBNNs. Additionally, since we explicitly predict kinetic and potential energies, we can use energy based-control, which has been demonstrated on a physical Furuta pendulum (Lutter et al., 2019a).

There are caveats to our study of SMM-Cs. We have only demonstrated SMM-Cs on robotic domains with smooth dynamics, while BBNNs have been demonstrated as useful model parameterizations on simulated contact-rich domains (Nagabandi et al., 2018; Chua et al., 2018; Nagabandi et al., 2019). However, SMMs can be used in contact-rich domains with the use of variational integrators (Manchester and Kuindersma, 2017). To do so, kinematic constraints resulting in contact are explicitly satisfied when making state predictions by solving a constrained optimization problem, where contact and non-holonomic constraints are enforced using Lagrange multipliers. Furthermore, constraints can be introduced and removed after the model is trained.

BBNNs used in contact-rich domains directly predict next-states (Nagabandi et al., 2018), as opposed to predicting state derivatives as presented in this work. The shortcoming of such an approach is that the kinematic constraints that result in contact can only be implicitly satisfied by minimizing prediction loss, with no guarantee that predictions made at test-time will satisfy the constraints. Additionally, if a new kinematic constraint is introduced, the BBNN will have to be re-trained. These properties warrant the study of SMMs in place of BBNNs for black-box modeling in contact-rich domains, and is an avenue for future work.

Another caveat worth mentioning is training and query time. We empirically find that it takes on the order of $5\times$ more CPU time per training epoch to train an SMM-C than to train a BBNN on the same amount of data. This is because more computation is required for SMM-C predictions compared to a BBNN. Some of this could be mitigated with various numerical and software optimizations.

Acknowledgments

We are grateful to Jeannette Bohg for advice. This work is supported in part by DARPA under agreement number D17AP00032. The content is solely the responsibility of the authors and does not necessarily represent the official views of DARPA.

References

Brian D. O. Anderson. *Optimal Control: Linear Quadratic Methods (Dover Books on Engineering)*. Dover Publications, 2007.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *arXiv preprint arXiv:1805.12114*, 2018.

Ken-ichi Funahashi and Yuichi Nakamura. Approximation of dynamical systems by continuous time recurrent neural networks. *Neural networks*, 6(6):801–806, 1993.

Taylor A Howell, Brian E Jackson, and Zachary Manchester. Altro: A fast solver for constrained trajectory optimization. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.

M. Kelly. An introduction to trajectory optimization: How to do your own direct collocation. *SIAM Review*, 59(4):849–904, 2017. doi: 10.1137/16M1062569.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1097–1105, 2012.

John G Kuschewski, Stefen Hui, and Stanislaw H Zak. Application of feedforward neural networks to dynamical system identification and control. *IEEE Transactions on Control Systems Technology*, 1(1):37–49, 1993.

Nathan O Lambert, Daniel S Drew, Joseph Yaconelli, Sergey Levine, Roberto Calandra, and Kristofer SJ Pister. Low-level control of a quadrotor with deep model-based reinforcement learning. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.

Michael Lutter, Kim Listmann, and Jan Peters. Deep Lagrangian networks for end-to-end learning of energy-based control for under-actuated systems. *arXiv preprint arXiv:1907.04489*, 2019a.

Michael Lutter, Christian Ritter, and Jan Peters. Deep Lagrangian networks: Using physics as model prior for deep learning. In *International Conference on Learning Representations*, 2019b.

Zachary Manchester and Scott Kuindersma. Variational contact-implicit trajectory optimization. In *International Symposium on Robotics Research (ISRR)*, 2017.

Jeremy Morton, Antony Jameson, Mykel J Kochenderfer, and Freddie Witherden. Deep dynamical modeling and control of unsteady fluid flows. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 9258–9268, 2018.

Richard M Murray, Zexiang Li, and S. Shankar Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC, 1994.

Anusha Nagabandi, Gregory Kahn, Ronald S Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 7559–7566, 2018.

Anusha Nagabandi, Kurt Konoglie, Sergey Levine, and Vikash Kumar. Deep Dynamics Models for Learning Dexterous Manipulation. In *Conference on Robot Learning (CoRL)*, 2019.

Duy Nguyen-Tuong and Jan Peters. Using model knowledge for learning inverse dynamics. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2677–2682, 2010.

Duy Nguyen-Tuong and Jan Peters. Model learning for robot control: a survey. *Cognitive processing*, 12(4):319–340, 2011.

Nathan Ratliff, Franziska Meier, Daniel Kappler, and Stefan Schaal. Doomed: Direct online optimization of modeling errors in dynamics. *Big Data*, 4(4):253–268, 2016.

C. Runge. Ueber die numerische auflösung von differentialgleichungen. *Mathematische Annalen*, 46(2):167–178, 1895.

Bruno Siciliano and Oussama Khatib. *Springer Handbook of Robotics*. Springer, 2008.

Mark W Spong. Energy based control of a class of underactuated mechanical systems. In *IFAC World Congress*, pages 431–435, 1996.