DISSIPATIVE SYMODEN: ENCODING HAMILTONIAN DYNAMICS WITH DISSIPATION AND CONTROL INTO DEEP LEARNING

Yaofeng Desmond Zhong[†*], **Biswadip Dey**[‡], **and Amit Chakraborty**[‡]
[†]Princeton University , [‡]Siemens Corporation, Corporate Technology
y.zhong@princeton.edu,(biswadip.dey,amit.chakraborty)@siemens.com

ABSTRACT

In this work, we introduce *Dissipative SymODEN*, a deep learning architecture which can infer the dynamics of a physical system with dissipation from observed state trajectories. To improve prediction accuracy while reducing network size, Dissipative SymODEN encodes the port-Hamiltonian dynamics with energy dissipation and external input into the design of its computation graph and learns the dynamics in a structured way. The learned model, by revealing key aspects of the system, such as the inertia, dissipation, and potential energy, paves the way for energy-based controllers.

1 INTRODUCTION

Inferring systems dynamics from observed trajectories plays a critical role in identification and control of complex, physical systems, such as robotic manipulators [1] and HVAC systems [2]. Although the use of neural networks in this context has a rich history of more than three decades [3], recent advances in deep learning [4] have led to renewed interest in this topic [5, 6, 7, 8, 9]. Deep neural networks learn underlying patterns from data and enable generalization beyond the training set by incorporating appropriate inductive bias into the learning approach. To promote representations that are *simple* in some sense, inductive bias [10, 11] often manifests itself via a set of assumptions and guides a learning algorithm to pick one hypothesis over another. The success in predicting an outcome for previously unseen data depends on how well the inductive bias captures the ground reality. Inductive bias can be introduced as the prior in a Bayesian model, or via the choice of computation graphs in a neural network.

Incorporation of physics-based priors into deep learning has been a key focus in the recent times. As these approaches use neural networks to approximate system dynamics, they are more expressive than traditional system identification techniques [12]. By using a directed graph to capture the causal relationships in a physical system, [13] introduces a recurrent graph network to infer latent space dynamics in robotic systems. [14] and [15] leveraged Lagrangian mechanics to learn the dynamics of kinematic structures from discrete observations. On the other hand, [16] and [17] have utilized Hamiltonian mechanics for learning dynamics from data. However, strict enforcement of the Hamiltonian prior is restrictive for real-life systems which often loses energy in a structured way (e.g. frictional losses in robotic arms, resistive losses in power grids, etc.).

To explicitly encode dissipation as a prior into end-to-end learning, we expand the scope of the Symplectic ODE-Net (SymODEN) architecture [17] and propose *Dissipative SymODEN*. The underlying dynamics is motivated by the port-Hamiltonian formulation [18], which has a correction term accounting for the prior of dissipation. With this term, Dissipative SymODEN can accommodate the energy losses from various sources of dissipation present in real-life systems. Our results show that inclusion of dissipation into the physics-informed SymODEN architecture improves its prediction accuracy and out-of-sample behavior, while offering insight about relevant physical properties of the system (such as inertia matrix, potential energy, energy dissipation etc.). These insights, in turn, can enable the use of energy-based controllers, such as the method of controlled Lagrangian [19] and interconnection & damping assignment [18], which offer performance guarantees for complex, nonlinear systems.

Contribution: The main contribution of this work is the introduction of a physics-informed learning architecture called Dissipative SymODEN which encodes a non-conservative physics, i.e. Hamiltonian dynamics with energy dissipation, into deep learning. This provides a means to uncover the dynamics of real-life physical systems whose Hamiltonian aspects have been adapted to external input and energy dissipation. By ensuring that the computation graph is aligned with the underlying physics, we achieve transparency, better predictions with smaller networks, and improved generalization. The architecture of Dissipative SymODEN has also been designed to accommodate angle data in the embedded form. Additionally, we use differentiable ODE solvers to avoid the need for derivative estimation.

[*]This research was conceived and primarily carried out during Y. D. Zhong's internship at Siemens Corporation, Corporate Technology. Post-internship, Y. D. Zhong's work was supported by ONR grant #N00014-18-1-2873.

2 THE PORT-HAMILTONIAN DYNAMICS

Hamiltonian dynamics is often used to systematically describe the dynamics of a physical system in the phase space $(\mathbf{q}, \mathbf{p})$, where $\mathbf{q} = (q_1, q_2, ..., q_n)$ is the generalized coordinate and $\mathbf{p} = (p_1, p_2, ..., p_n)$ is the generalized momentum. In this approach, the key to the dynamics is a scalar function $H(\mathbf{q}, \mathbf{p})$, which is referred to as the Hamiltonian. In almost all physical systems, the Hamiltonian represents the total energy which can be expressed as

$$H(\mathbf{q}, \mathbf{p}) = \frac{1}{2}\mathbf{p}^T \mathbf{M}^{-1}(\mathbf{q})\mathbf{p} + V(\mathbf{q}), \tag{1}$$

where $\mathbf{M}(\mathbf{q})$ is the symmetric positive definite mass/inertia matrix and $V(\mathbf{q})$ represents the potential energy of the system. The equations of motion are governed by the symplectic gradient [20] of the Hamiltonian, i.e.,

$$\dot{\mathbf{q}} = \frac{\partial H}{\partial \mathbf{p}} \qquad \dot{\mathbf{p}} = -\frac{\partial H}{\partial \mathbf{q}}. \tag{2}$$

Moreover, since $\dot{H} = (\frac{\partial H}{\partial \mathbf{q}})^T \dot{\mathbf{q}} + (\frac{\partial H}{\partial \mathbf{p}})^T \dot{\mathbf{p}} = 0$, moving along the symplectic gradient conserves the Hamiltonian (i.e. the total energy). However, although the classical Hamiltonian dynamics ensures energy conservation, it fails to model dissipation and external inputs, which often appear in real-life systems. The port-Hamiltonian dynamics generalizes the classical Hamiltonian dynamics by explicitly modelling the total energy, dissipation and external inputs. Motivated by this formulation, we consider the following port-Hamiltonian dynamics in this work:

$$\begin{bmatrix} \dot{\mathbf{q}} \\ \dot{\mathbf{p}} \end{bmatrix} = \left(\begin{bmatrix} \mathbf{0} & \mathbf{I} \\ -\mathbf{I} & \mathbf{0} \end{bmatrix} - \mathbf{D}(\mathbf{q}) \right) \begin{bmatrix} \frac{\partial H}{\partial \mathbf{q}} \\ \frac{\partial H}{\partial \mathbf{p}} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{g}(\mathbf{q}) \end{bmatrix} \mathbf{u}, \tag{3}$$

where the dissipation matrix $\mathbf{D}(\mathbf{q})$ is symmetric positive semi-definite and represents energy dissipation. The external input $\mathbf{u}$ is usually affine and only affects the generalized momenta. The input matrix $\mathbf{g}(\mathbf{q})$ is assumed to have full column rank. As expected, with zero dissipation and zero input, (3) reduces to the classical Hamiltonian dynamics.

3 DISSIPATIVE SYMPLECTIC ODE-NET

3.1 TRAINING NEURAL ODE WITH CONSTANT FORCING

We focus on the problem of learning an ordinary differential equation (ODE) from observation data. Assume the analytical form of the right hand side (RHS) of an ODE "$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u})$" is unknown. An observation data $\mathbf{X} = ((\mathbf{x}_{t_0}, \mathbf{u}_c), ..., (\mathbf{x}_{t_n}, \mathbf{u}_c))$ with a constant input $\mathbf{u}_c$ allows us to approximate $\mathbf{f}(\mathbf{x}, \mathbf{u})$ with a neural net by leveraging

$$\left. \begin{bmatrix} \dot{\mathbf{x}} \\ \dot{\mathbf{u}} \end{bmatrix} \right|_{\mathbf{X}} = \begin{bmatrix} \mathbf{f}_\theta(\mathbf{x}, \mathbf{u}) \\ \mathbf{0} \end{bmatrix} = \tilde{\mathbf{f}}_\theta(\mathbf{x}, \mathbf{u}). \tag{4}$$

Equation (4), by matching the input and output dimensions, enables us to feed it into Neural ODE [21]. With Neural ODE, we make predictions by approximating the RHS of (4) using a neural network and feed it into an ODE solver

$$(\hat{\mathbf{x}}, \mathbf{u}_c)_{t_1, t_2, ..., t_n} = \text{ODESolve}((\mathbf{x}, \mathbf{u}_c)_{t_0}, \tilde{\mathbf{f}}_\theta, t_1, t_2, ..., t_n).$$

We can then construct the loss function $L = \|\mathbf{X} - \hat{\mathbf{X}}\|_2^2$. In practice, we introduce the time horizon τ as a hyperparameter and predict $\mathbf{x}_{t_{i+1}}, \mathbf{x}_{t_{i+2}}, ..., \mathbf{x}_{t_{i+\tau}}$ from initial condition $\mathbf{x}_{t_i}$, where $i = 0, ..., n - \tau$. The problem is then how to design the network architecture of $\tilde{\mathbf{f}}_\theta$, or equivalently $\mathbf{f}_\theta$.

3.2 LEARNING FROM GENERALIZED COORDINATE AND MOMENTUM

Suppose we have data consisting of $(\mathbf{q}, \mathbf{p}, \mathbf{u})_{t_0, ..., t_n}$, where $\mathbf{u}$ remains constant in each trajectory. We use four neural nets – $\mathbf{M}_{\theta_1}^{-1}(\mathbf{q})$, $V_{\theta_2}(\mathbf{q})$, $\mathbf{g}_{\theta_3}(\mathbf{q})$ and $\mathbf{D}_{\theta_4}(\mathbf{q})$ – as function approximators to represent the inverse of mass matrix, potential energy, the input matrix and the dissipation matrix, respectively. Thus,

$$\mathbf{f}_\theta(\mathbf{q}, \mathbf{p}, \mathbf{u}) = \left(\begin{bmatrix} \mathbf{0} & \mathbf{I} \\ -\mathbf{I} & \mathbf{0} \end{bmatrix} - \mathbf{D}_{\theta_4}(\mathbf{q}) \right) \begin{bmatrix} \frac{\partial H_{\theta_1, \theta_2}}{\partial \mathbf{q}} \\ \frac{\partial H_{\theta_1, \theta_2}}{\partial \mathbf{p}} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{g}_{\theta_3}(\mathbf{q}) \end{bmatrix} \mathbf{u} \tag{5}$$

where

$$H_{\theta_1, \theta_2}(\mathbf{q}, \mathbf{p}) = \frac{1}{2}\mathbf{p}^T \mathbf{M}_{\theta_1}^{-1}(\mathbf{q})\mathbf{p} + V_{\theta_2}(\mathbf{q}) \tag{6}$$

The partial derivative can be taken care of by automatic differentiation. By putting the designed $\mathbf{f}_\theta(\mathbf{q}, \mathbf{p}, \mathbf{u})$ into Neural ODE, we obtain a systematic way of adding the prior knowledge of a structured dynamics into end-to-end learning.

3.3 Learning from Embedded Angle Data

Often, especially in robotics, the state variables involve angles residing in the interval $[-\pi, \pi)$. In other words, each angle lies on the manifold $\mathbb{S}^1$. However, generalized coordinates are typically assumed to lie on $\mathbb{R}^n$. To bridge this gap, we use an angle-aware design [17] and assume that the generalized coordinates are angles available as $(\mathbf{x}_1(\mathbf{q}), \mathbf{x}_2(\mathbf{q}), \mathbf{x}_3(\dot{\mathbf{q}}), \mathbf{u})_{t_0,\ldots,t_n} = (\cos\mathbf{q}, \sin\mathbf{q}, \dot{\mathbf{q}}, \mathbf{u})_{t_0,\ldots,t_n}$. Then, similar to [17], we aim to learn a structured dynamics (3) expressed in terms of $\mathbf{x}_1$, $\mathbf{x}_2$ and $\mathbf{x}_3$. As $\mathbf{p} = \mathbf{M}(\mathbf{x}_1, \mathbf{x}_2)\dot{\mathbf{q}}$, we can express this dynamics as

$$\begin{aligned}
\dot{\mathbf{x}}_1 &= -\sin\mathbf{q} \circ \dot{\mathbf{q}} = -\mathbf{x}_2 \circ \dot{\mathbf{q}} \\
\dot{\mathbf{x}}_2 &= \cos\mathbf{q} \circ \dot{\mathbf{q}} = \mathbf{x}_1 \circ \dot{\mathbf{q}} \\
\dot{\mathbf{x}}_3 &= \frac{\mathrm{d}}{\mathrm{d}t}(\mathbf{M}^{-1}(\mathbf{x}_1, \mathbf{x}_2)\mathbf{p}) = \frac{\mathrm{d}}{\mathrm{d}t}(\mathbf{M}^{-1}(\mathbf{x}_1, \mathbf{x}_2))\mathbf{p} + \mathbf{M}^{-1}(\mathbf{x}_1, \mathbf{x}_2)\dot{\mathbf{p}},
\end{aligned} \tag{7}$$

where "$\circ$" represents the element-wise product. We assume $\mathbf{q}$ and $\mathbf{p}$ evolve with the structured dynamics Equation (3) and substitute Equation (3) in to the RHS of Equation (7). Similar to our approach in Sec 3.2, we use four neural nets to express the RHS of Equation (7) as $\mathbf{f}_\theta(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{u})$. Thus, it can be fed into Equation 4 and the Neural ODE.

3.4 Learning on Hybrid Spaces $\mathbb{R}^n \times \mathbb{T}^m$

In most of physical systems, both translational coordinates and rotational coordinates coexist. In other words, the generalized coordinates lie on $\mathbb{R}^n \times \mathbb{T}^m$, where $\mathbb{T}^m$ denotes the m-torus. Here we put together the architecture of the previous two subsections. We assume the generalized coordinates are $\mathbf{q} = (\mathbf{r}, \boldsymbol{\phi}) \in \mathbb{R}^n \times \mathbb{T}^m$ and the data comes in the form of $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5, \mathbf{u})_{t_0,\ldots,t_n} = (\mathbf{r}, \cos\boldsymbol{\phi}, \sin\boldsymbol{\phi}, \dot{\mathbf{r}}, \dot{\boldsymbol{\phi}}, \mathbf{u})_{t_0,\ldots,t_n}$. We use four neural nets – $\mathbf{M}^{-1}_{\theta_1}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$, $V_{\theta_2}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$, $\mathbf{g}_{\theta_3}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$ and $\mathbf{D}_{\theta_4}(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$ – as function approximators. Then the dynamics is given by

$$[\dot{\mathbf{x}}_1, \dot{\mathbf{x}}_2, \dot{\mathbf{x}}_3, \dot{\mathbf{x}}_4, \dot{\mathbf{x}}_5]^T = \mathbf{f}_\theta(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_5, \mathbf{u})$$

3.5 The Dissipation Matrix and the Mass matrix

As the dissipation matrix models energy dissipation such as friction and resistance, it is positive semi-definite. We impose this constraint in the network architecture by $\mathbf{D}_{\theta_4} = \mathbf{L}_{\theta_4}\mathbf{L}^T_{\theta_4}$, where $\mathbf{L}_{\theta_4}$ is a lower-triangular matrix. In real physical systems, both the mass matrix $\mathbf{M}$ and its inverse are positive definite. Similarly, semi-definiteness is constraint by $\mathbf{M}^{-1}_{\theta_1} = \mathbf{L}_{\theta_1}\mathbf{L}^T_{\theta_1}$, where $\mathbf{L}_{\theta_1}$ is a lower-triangular matrix. The positive definiteness is ensured by adding a small constant ϵ to the diagonal elements of $\mathbf{M}^{-1}_{\theta_1}$. It not only makes $\mathbf{M}_{\theta_1}$ invertible, but also stabilizes training.

4 Experiments

4.1 Experimental Setup

We use the following four tasks to evaluate the performance of Dissipative SymODEN architecture – (i) Task 1: a pendulum with generalized coordinate and momentum data; (ii) Task 2: a pendulum with embedded angle data; (iii) Task 3: a CartPole system; and (iv) Task 4: an Acrobot.

Model Variants: Besides the Dissipative SymODEN model derived above, we consider a variant, called *Unstructured (Unstr.) Dissipative SymODEN*, which approximates the Hamiltonian by a fully connected neural net H_{θ_1,θ_2}. We also consider the original *SymODEN* [17] as a model variant.

Baseline Models: We set up baseline models for all four experiments. For the pendulum with generalized coordinate and momentum data, the *naive baseline* model approximates (5) – $\mathbf{f}_\theta(\mathbf{x}, \mathbf{u})$ – by a fully connected neural net. For all the other experiments, which involves embedded angle data, we set up two different baseline models: *naive baseline* approximates $\mathbf{f}_\theta(\mathbf{x}, \mathbf{u})$ by a fully connected neural net. Also, we set up the *geometric baseline* model which approximates $\dot{\mathbf{q}}$ and $\dot{\mathbf{p}}$ with a fully connected neural net.

Data Generation: For all tasks, we randomly generated initial conditions of states and subsequently combined them with 5 different constant control inputs, i.e., $u = -2.0, -1.0, 0.0, 1.0, 2.0$, to produce the initial conditions and input required for simulation. The simulators integrate the corresponding dynamics for 20 time steps to generate trajectory data which is then used to construct the training set and test set.

Model training: In all the tasks, we train our model using Adam optimizer [22] with 1000 epochs. We set a time horizon $\tau = 3$, and choose "RK4" as the numerical integration scheme in Neural ODE. We logged the *train error*, *test error* and *prediction (pred.) error* per trajectory for all the tasks. Prediction error per trajectory is calculated by using the same initial state condition in the training set with a constant control of $u = 0.0$, integrating 40 time steps forward.

4.2 Task 1: Pendulum with Generalized Coordinate and Momentum Data

In this task, we use the model described in Section 3.2 and present the predicted trajectories of the learned models as well as the learned functions of Dissipative SymODEN. The underlying dynamics is given by

$$\dot{q} = 3p, \qquad \dot{p} = -5\sin q - 0.3p + u, \quad (8)$$

with the Hamiltonian $H(q,p) = 1.5p^2 + 5(1-\cos q)$. In other words $M^{-1}(q) = 3$, $V(q) = 5(1-\cos q)$, $g(q) = 1$ and $\mathbf{D}_{\theta_4}(q) = [0, 0; 0, 0.1]$. Figure 1 shows that the learned $g_{\theta_3}(q)$ and $M^{-1}_{\theta_1}(q)$ matches the ground truth pretty well. Also, $V_{\theta_2}(q)$ differs from the ground truth by an almost constant margin which is expected since only the derivative of $V_{\theta_2}(q)$ impacts the dynamics. The learned dissipation matrix $\mathbf{D}_{\theta_4}(q)$ does not match the ground truth. We address this issue in the next subsection. In Table 1, Naive Baseline's prediction error is the lowest because predicted trajectories reach the origin faster than the ground truth.

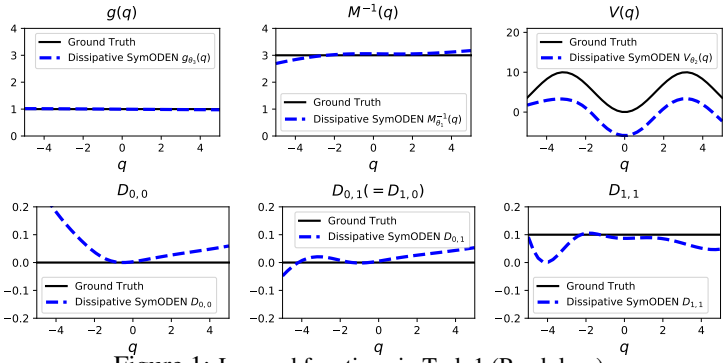

Figure 1: Learned functions in Task 1 (Pendulum).

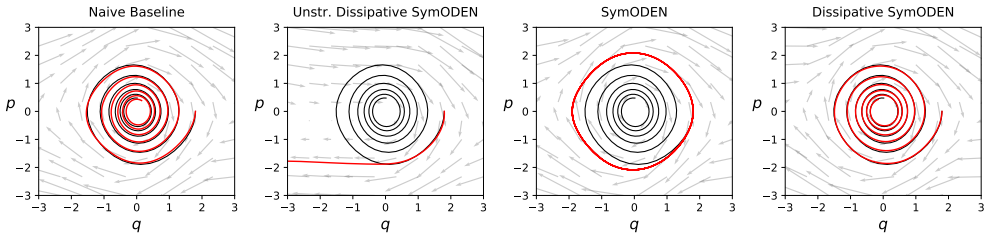

Figure 2: Learned trajectories of different models. Red and black lines represent the learned and ground truth trajectories, respectively and the gray arrows show the vector fields learned by each model. *Dissipative SymODEN* learns a more accurate vector field than the *naive baseline* model. Moreover, it appears that whereas *SymODEN* learns an energy-conserved vector field slightly different from the ground truth, *Unstructured Dissipative SymODEN* learns it completely wrong.

4.3 Task 2: Pendulum with Embedded Data

In this task, the dynamics is the same as Equation (8) but the training data are generated by the OpenAI Gym simulator, i.e. we use embedded angle data and assume we only have access to $\dot{q}$ instead of p. We use the model described in Section 3.3 to learn the structured dynamics. Without true p data, the learned function matches the ground truth with a scaling β, as shown in Figure 3. Please refer to [17] for explanation of the scaling. In this example, with the scaling $\beta = 0.357$, the learned functions match the ground truth. With the angle-aware design, we learned the dissipation matrix much better than the previous subsection.

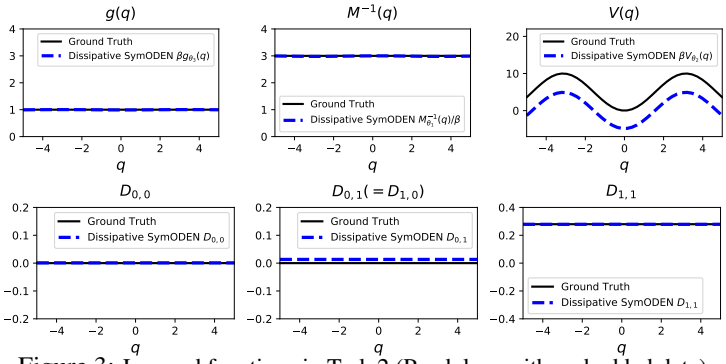

Figure 3: Learned functions in Task 2 (Pendulum with embedded data).

4.4 Results

In Table 1, we show the train, test and prediction errors for all four tasks. Dissipative SymODEN performs the best in all three metrics. As SymODEN does not allow dissipation, it does not perform well in these tasks. Since Unstructured Dissipative SymODEN architecture has trouble learning a good vector field, it performs the worst in all the tasks except Task 2. In conclusion, Dissipative SymODEN achieves higher accuracy with less model parameters. Moreover, the learned model reveals physical aspects of the system, which can be leveraged by energy-based controllers.

References

[1] 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.

[2] T. Wei, Y. Wang, and Q. Zhu, "Deep Reinforcement Learning for Building HVAC Control," in *Proceedings of the 54th Annual Design Automation Conference (DAC)*, 2017, pp. 22:1–22:6.

Table 1: Train, Test and Prediction Errors of Four Tasks

Task		Naive Baseline	Geometric Baseline	UnStr. Dissipative SymODEN	SymODEN	Dissipative SymODEN
1	*#Parameters*	0.36M	N/A	0.22M	0.13M	0.15M
	Train error	26.38 ± 38.00	N/A	34.80 ± 68.53	4.47 ± 6.40	0.88 ± 1.41
	Test error	35.03 ± 49.89	N/A	49.44 ± 81.31	7.52 ± 10.13	1.25 ± 1.81
	Pred. error	32.544 ± 36.203	N/A	219.36 ± 296.86	96.50 ± 99.56	34.03 ± 47.83
2	*#Parameters*	0.65M	0.46M	0.41M	0.14M	0.16M
	Train error	2.02 ± 4.41	0.42 ± 1.16	1.90 ± 3.85	2.37 ± 2.71	0.15 ± 0.27
	Test error	2.01 ± 4.99	0.33 ± 1.22	1.61 ± 3.36	2.67 ± 2.83	0.13 ± 0.25
	Pred. error	40.18 ± 78.10	0.81 ± 0.68	7.04 ± 13.65	72.78 ± 90.42	1.04 ± 1.3
3	*#Parameters*	1.01M	0.82M	0.69M	0.51M	0.53M
	Train error	12.92 ± 15.58	0.48 ± 0.50	12.09 ± 18.38	3.33 ± 3.85	0.88 ± 0.89
	Test error	20.07 ± 26.42	1.34 ± 3.19	19.87 ± 23.16	3.80 ± 3.71	1.37 ± 1.30
	Pred. error	268.24 ± 204.15	60.12 ± 96.18	366.38 ± 405.45	30.21 ± 34.33	8.32 ± 7.81
4	*#Parameters*	1.46M	0.97M	0.80M	0.51M	0.53M
	Train error	1.76 ± 2.26	1.90 ± 2.82	77.56 ± 111.50	2.92 ± 2.58	0.47 ± 0.64
	Test error	5.12 ± 9.14	4.87 ± 7.42	122.70 ± 190.90	5.27 ± 6.55	0.81 ± 1.10
	Pred. error	36.65 ± 77.16	44.26 ± 95.70	590.77 ± 807.88	68.26 ± 103.46	12.72 ± 32.12

[3] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Transactions on Neural Networks*, vol. 1, no. 1, pp. 4–27, 1990.

[4] I. Goodfellow, A. Courville, and Y. Bengio, *Deep learning*. MIT Press, 2016, vol. 1.

[5] M. Watter, J. Springenberg, J. Boedecker, and M. Riedmiller, "Embed to control: A locally linear latent dynamics model for control from raw images," in *Advances in Neural Information Processing 29*, 2015, pp. 2746–2754.

[6] M. Karl, M. Soelch, J. Bayer, and P. van der Smagt, "Deep variational bayes filters: Unsupervised learning of state space models from raw data," *arXiv:1605.06432*, 2016.

[7] R. G. Krishnan, U. Shalit, and D. Sontag, "Structured inference networks for nonlinear state space models," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[8] A. Byravan and D. Fox, "Se3-nets: Learning rigid body motion using deep neural networks," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 173–180.

[9] I. Ayed, E. de Bézenac, A. Pajot, J. Brajard, and P. Gallinari, "Learning dynamical systems from partial observations," *arXiv:1902.11136*, 2019.

[10] D. Haussler, "Quantifying inductive bias: AI learning algorithms and Valiant's learning framework," *Artificial Intelligence*, vol. 36, no. 2, pp. 177–221, 1988.

[11] J. Baxter, "A model of inductive bias learning," *Journal of Artificial Intelligence Research*, vol. 12, pp. 149–198, 2000.

[12] T. Söderström and P. Stoica, *System identification*. Prentice-Hall, Inc., 1988.

[13] A. Sanchez-Gonzalez, N. Heess, J. T. Springenberg, J. Merel, M. Riedmiller, R. Hadsell, and P. Battaglia, "Graph networks as learnable physics engines for inference and control," in *International Conference on Machine Learning (ICML)*, 2018, pp. 4467–4476.

[14] M. Lutter, C. Ritter, and J. Peters, "Deep lagrangian networks: Using physics as model prior for deep learning," in *7th International Conference on Learning Representations (ICLR)*, 2019.

[15] J. K. Gupta, K. Menda, Z. Manchester, and M. J. Kochenderfer, "A general framework for structured learning of mechanical systems," *arXiv:1902.08705*, 2019.

[16] S. Greydanus, M. Dzamba, and J. Yosinski, "Hamiltonian Neural Networks," *arXiv:1906.01563*, 2019.

[17] Y. D. Zhong, B. Dey, and A. Chakraborty, "Symplectic ODE-Net: Learning Hamiltonian Dynamics with Control," in *International Conference on Learning Representations (ICLR)*, 2020.

[18] R. Ortega, A. J. Van Der Schaft, B. Maschke, and G. Escobar, "Interconnection and damping assignment passivity-based control of port-controlled hamiltonian systems," *Automatica*, vol. 38, no. 4, pp. 585–596, 2002.

[19] A. M. Bloch, N. E. Leonard, and J. E. Marsden, "Controlled lagrangians and the stabilization of euler–poincaré mechanical systems," *International Journal of Robust and Nonlinear Control*, vol. 11, no. 3, pp. 191–214, 2001.

[20] D. J. Rowe, A. Ryman, and G. Rosensteel, "Many-body quantum mechanics as a symplectic dynamical system," *Physical Review A*, vol. 22, no. 6, p. 2362, 1980.

[21] T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," in *Advances in Neural Information Processing Systems 31*, 2018, pp. 6571–6583.

[22] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv:1412.6980*, 2014.