Benchmarking Energy-Conserving Neural Networks for Learning Dynamics from Data

Yaofeng Desmond Zhong   YAOFENG.ZHONG@SIEMENS.COM
Biswadip Dey   BISWADIP.DEY@SIEMENS.COM
Amit Chakraborty   AMIT.CHAKRABORTY@SIEMENS.COM
Siemens Corporation, 755 College Road East, Princeton, NJ 08536

Abstract

The last few years have witnessed an increased interest in incorporating physics-informed inductive bias in deep learning frameworks. In particular, a growing volume of literature has been exploring ways to enforce energy conservation while using neural networks for learning dynamics from observed time-series data. In this work, we survey ten recently proposed energy-conserving neural network models, including HNN, LNN, DeLaN, SymODEN, CHNN, CLNN and their variants. We provide a compact derivation of the theory behind these models and explain their similarities and differences. Their performance are compared in 4 physical systems. We point out the possibility of leveraging some of these energy-conserving models to design energy-based controllers.

Keywords: Inductive Bias, Deep Learning, Physics-based Priors, Neural ODE

1. Introduction

The motion of physical systems is governed by the laws of physics, which can be expressed as a set of differential equations. These equations describe how the state of the system, e.g., the position and velocity of objects, evolve over time. Given such a set of differential equations and an initial state, we can solve the initial value problem for future states and predict the trajectory of the physical system. On the other hand, if the underlying dynamics, i.e., the differential equations, are unknown and a set of trajectory data is given instead, we can formulate a learning problem to infer the unknown dynamics that can explain the trajectory data and leverage the learned differential equations for making predictions based on a set of new initial states. From a learning perspective, deep neural networks are very effective function approximators (Pinkus, 1999; Liang and Srikant, 2017) and recent advances in end-to-end representation learning have enabled their wide adoption in various application domains (Goodfellow et al., 2016). To avoid overfitting and enable generalization beyond training data, the neural networks employ appropriate inductive bias by preferring certain representations over others (Haussler, 1988); often, the computation graph is constructed to capture/reflect some salient aspects of the problem under consideration.

In recent years, a growing number of neural network models have been proposed to solve the problem of learning dynamics, expressed via a set of differential equations, from data (Lutter et al., 2019b; Greydanus et al., 2019; Zhong et al., 2020a; Chen et al., 2020; Roehrl et al., 2020; Cranmer et al., 2020; Finzi et al., 2020). Most of these models are based on Neural ODE (Chen et al., 2018), a neural network approach to learning an unknown set of differential equations from trajectory data. When it comes to learning the motion of physical systems, we can leverage the laws of physics for data efficiency and generalization. Two popular choices to model the relevant physics

are the Lagrangian and Hamiltonian dynamics, which are both reformulations of Newton's second law and can model a broad class of physical systems. Moreover, both of these formalisms conserve energy under proper assumptions, which serves as a great physics-based prior when modeling energy-conserved system. In addition, there is a growing volume of work that focuses on enforcing physics while learning the underlying dynamics from high-dimensional image sequence or video data (Zhong and Leonard, 2020; Allen-Blanchette et al., 2020).

In this work, we benchmark ten recently introduced neural network models that impose structures on the underlying computation graph for learning Lagrangian or Hamiltonian dynamics from trajectory data. To aid informed selection of neural network model(s) for a given task, we focus on highlighting and explaining the similarities and differences between these models in 4 physical systems.

2. Preliminaries

Lagrangian and Hamiltonian dynamics are both reformulations of Newton's second law of motion. If holonomic constraints exist, those constraints can be enforced either implicitly or explicitly. The former is simpler to derive, while the latter simplifies the functional form. In this section, we review Lagrangian and Hamiltonian dynamics with both implicit and explicit constraints, derived from D'Alembert's principle (the principle of virtual work). Please see Goldstein et al. (2002) for more details on the implicit constraint derivation, and Date (2010), Finzi et al. (2020) for the explicit constraint derivation using variational principle. We wrap up this section by reviewing how to learn these dynamics from data.

2.1. Lagrangian dynamics with implicit constraints

For a rigid body system with degrees of freedom (DOF) M, the configuration of rigid bodies at time t can be described by independent generalized coordinates $\mathbf{q}(t) = (q_1(t), q_2(t), ..., q_M(t))$. From D'Alembert's principle (the principle of virtual work), Newton's second law can be expressed as

$$\delta \mathbf{q}^T \Big[\frac{\mathrm{d}}{\mathrm{d}t} \Big(\frac{\partial L}{\partial \dot{\mathbf{q}}} \Big) - \frac{\partial L}{\partial \mathbf{q}} \Big] = 0, \tag{1}$$

where the scalar function $L(\mathbf{q}, \dot{\mathbf{q}})$ is the Lagrangian, $\dot{\mathbf{q}} = \mathrm{d}\mathbf{q}/\mathrm{d}t$ and $\delta\mathbf{q}$ is the virtual displacement. Here we assume all the generalized forces are conservative which can be captured by a potential field. As the M generalized coordinates are independent, each δq_i can be varied independently. Then the only way for Equation (1) to hold is the following set of M equations,

$$\frac{\mathrm{d}}{\mathrm{d}t} \Big(\frac{\partial L}{\partial \dot{\mathbf{q}}} \Big) - \frac{\partial L}{\partial \mathbf{q}} = \mathbf{0}, \tag{2}$$

which is also referred to as the Euler-Lagrange equation. We can symbolically expand the Euler-Lagrange equation into a set of M second-order differential equations,

$$\ddot{\mathbf{q}} = (\nabla_{\dot{\mathbf{q}}} \nabla_{\dot{\mathbf{q}}}^T L)^{-1} [\nabla_{\mathbf{q}} L - (\nabla_{\mathbf{q}} \nabla_{\dot{\mathbf{q}}}^T L) \dot{\mathbf{q}}], \tag{3}$$

where the operators are defined as $(\nabla_{\dot{\mathbf{q}}})_i = \frac{\partial}{\partial \dot{q}_i}$, $(\nabla_{\mathbf{q}} \nabla_{\dot{\mathbf{q}}}^T)_{ij} = \frac{\partial^2}{\partial q_j \partial \dot{q}_i}$ and $(\nabla_{\dot{\mathbf{q}}} \nabla_{\dot{\mathbf{q}}}^T)_{ij} = \frac{\partial^2}{\partial \dot{q}_j \partial \dot{q}_i}$.

The physical meaning of the Lagrangian $L(\mathbf{q}, \dot{\mathbf{q}})$ is the difference between kinetic energy $T(\mathbf{q}, \dot{\mathbf{q}})$ and potential energy $V(\mathbf{q})$. For rigid body systems, the Lagrangian is

$$L(\mathbf{q}, \dot{\mathbf{q}}) = T(\mathbf{q}, \dot{\mathbf{q}}) - V(\mathbf{q}) = \frac{1}{2}\dot{\mathbf{q}}^T\mathbf{M}(\mathbf{q})\dot{\mathbf{q}} - V(\mathbf{q}), \tag{4}$$

where $\mathbf{M}(\mathbf{q})$ is the positive definite mass matrix. With this form, Equation (3) can be written as

$$\ddot{\mathbf{q}} = \mathbf{M}(\mathbf{q})^{-1}[-\frac{1}{2}\nabla_{\mathbf{q}}\Big(\dot{\mathbf{q}}^T\mathbf{M}(\mathbf{q})\dot{\mathbf{q}}\Big) - \nabla_{\mathbf{q}}V(\mathbf{q})]. \tag{5}$$

2.2. Hamiltonian dynamics with implicit constraints

The Hamiltonian dynamics can be derived by performing Legendre transformation on the Lagrangian to get the conjugate momentum $\mathbf{p} = \nabla_{\dot{\mathbf{q}}}L$ and the Hamiltonian $H(\mathbf{q}, \mathbf{p}) = \dot{\mathbf{q}}^T\mathbf{p} - L(\mathbf{q}, \dot{\mathbf{q}})$. The Euler-Lagrangian equation (2) can then be expressed as the following Hamiltonian dynamics

$$\dot{\mathbf{q}} = \nabla_{\mathbf{p}}H, \qquad \dot{\mathbf{p}} = -\nabla_{\mathbf{q}}H. \tag{6}$$

Along a trajectory of Hamiltonian dynamics, the Hamiltonian H is conserved, since

$$\dot{H} = (\nabla_{\mathbf{q}}^T H)\dot{\mathbf{q}} + (\nabla_{\mathbf{p}}^T H)\dot{\mathbf{p}} = 0. \tag{7}$$

For a rigid body system, we can use the structured Lagrangian in Equation (4) and get the conjugate momentum $\mathbf{p} = \mathbf{M}(\mathbf{q})\dot{\mathbf{q}}$ and the structured Hamiltonian

$$H(\mathbf{q}, \mathbf{p}) = \frac{1}{2}\mathbf{p}^T\mathbf{M}^{-1}(\mathbf{q})\mathbf{p} + V(\mathbf{q}), \tag{8}$$

The Hamiltonian equals the total energy of the rigid body system and Equation (7) shows the energy conservation property of Hamiltonian dynamics. With the structured Hamiltonian (8), we can express the Hamiltonian dynamics as

$$\dot{\mathbf{q}} = \mathbf{M}^{-1}(\mathbf{q})\mathbf{p}, \qquad \dot{\mathbf{p}} = -\frac{1}{2}\nabla_{\mathbf{q}}\Big(\mathbf{p}^T\mathbf{M}^{-1}(\mathbf{q})\mathbf{p}\Big) - \nabla_{\mathbf{q}}V(\mathbf{q}) \tag{9}$$

2.3. Lagrangian dynamics with explicit constraints

The derivations above enforce holonomic constraints implicitly by choosing a number of generalized coordinates that equals the degrees of freedom M. To derive Lagrangian and Hamiltonian dynamics, we can also enforce the holonomic constraints explicitly so that we can use Cartesian coordinates to describe the configuration of the rigid body system. Finzi et al. (2020) shows that choosing Cartesian coordinates and enforcing constraints explicitly simplifies the functional form of Hamiltonian and Lagrangian, which facilitates learning. Here we show a derivation based on D'Alembert's principle.

Assume the configuration of rigid bodies at time t can be described by a set of Cartesian coordinates $\mathbf{x}(t) = (x_1(t), x_2(t), ..., x_D(t))$. From D'Alembert's principle, with a Lagrangian $L(\mathbf{x}, \dot{\mathbf{x}})$, Newton's second law can be expressed as

$$\delta\mathbf{x}^T\Big[\frac{\mathrm{d}}{\mathrm{d}t}\Big(\frac{\partial L}{\partial\dot{\mathbf{x}}}\Big) - \frac{\partial L}{\partial\mathbf{x}}\Big] = 0, \tag{10}$$

with the Lagrangian,

$$L(\mathbf{x}, \dot{\mathbf{x}}) = \frac{1}{2}\dot{\mathbf{x}}^T \mathbf{M}\dot{\mathbf{x}} - V(\mathbf{x}), \tag{11}$$

where the mass matrix $\mathbf{M} = \nabla_{\dot{\mathbf{x}}}\nabla_{\dot{\mathbf{x}}}^T L$ is a constant matrix that does not depend on the coordinates, which simplifies the functional form of Lagrangian. Please see Finzi et al. (2020) for more details. Since the x_is are not independent, we need to incorporate the relationship between the x_is - the constraints - to set up the equations of motion. In general, for a system of DOF M, there exists $D - M$ constraints. In this work, we assume all the constraints are holonomic, thus can be expressed as $K = D - M$ equality constraints $\Phi_k(\mathbf{x}) = 0$ for $k = 1, 2, ..., K$. We collect functions $\Phi_k(\mathbf{x})$ into a column vector $\Phi(\mathbf{x})$ so the virtual displacements δx_is are related by K equations $(D_{\mathbf{x}}\Phi)\delta\mathbf{x} = \mathbf{0}$, where $D_{\mathbf{x}}\Phi$ denotes the Jacobian. We introduce K Lagrangian multipliers λ_ks collected as a column vector $\lambda_L \in \mathbb{R}^K$ and we have $\lambda_L^T(D_{\mathbf{x}}\Phi)\delta\mathbf{x} = 0$. Adding this scalar to Equation (10), we get

$$\delta\mathbf{x}^T\left[\frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L}{\partial\dot{\mathbf{x}}}\right) - \frac{\partial L}{\partial\mathbf{x}} + (D_{\mathbf{x}}\Phi)^T\lambda_L\right] = 0. \tag{12}$$

Even though the δx_is are not independent, we can choose Lagrangian multipliers λ_L such that the only solution to Equation (12) is one where the coefficients of all δx_is are zeros. We symbolically expand the coefficients and get

$$\mathbf{M}\ddot{\mathbf{x}} + \nabla_{\mathbf{x}}V + (D_{\mathbf{x}}\Phi)^T\lambda_L = \mathbf{0}. \tag{13}$$

Note here we use the fact that $\nabla_{\mathbf{x}}\nabla_{\dot{\mathbf{x}}}^T L = \mathbf{0}$ and $\nabla_{\mathbf{x}}L = -\nabla_{\mathbf{x}}V$. The Lagrangian multipliers λ_L can be solved as functions of $\mathbf{x}$ and $\dot{\mathbf{x}}$ by leveraging $\dot{\Phi} = (D_{\mathbf{x}}\Phi)\dot{\mathbf{x}} = \mathbf{0}$ and $\ddot{\Phi} = (D_{\mathbf{x}}\Phi)\ddot{\mathbf{x}} + (D_{\mathbf{x}}\dot{\Phi})\dot{\mathbf{x}} = \mathbf{0}$ to eliminate $\ddot{\mathbf{x}}$. The solution is

$$\lambda_L = -[(D_{\mathbf{x}}\Phi)\mathbf{M}^{-1}(D_{\mathbf{x}}\Phi)^T]^{-1}[(D_{\mathbf{x}}\Phi)\mathbf{M}^{-1}\nabla_{\mathbf{x}}V - (D_{\mathbf{x}}\dot{\Phi})\dot{\mathbf{x}}] \tag{14}$$

Substitute the solved λ_L into Equation (13), we get a set of D second-order differential equations,

$$\ddot{\mathbf{x}} = \mathbf{M}^{-1}(D_{\mathbf{x}}\Phi)^T[(D_{\mathbf{x}}\Phi)\mathbf{M}^{-1}(D_{\mathbf{x}}\Phi)^T]^{-1}[(D_{\mathbf{x}}\Phi)\mathbf{M}^{-1}\nabla_{\mathbf{x}}V - (D_{\mathbf{x}}\dot{\Phi})\dot{\mathbf{x}}] - \mathbf{M}^{-1}\nabla_{\mathbf{x}}V. \tag{15}$$

2.4. Hamiltonian dynamics with explicit constraints

By applying Legendre transformation on the Lagrangian (11) we get the conjugate momentum $\mathbf{p_x} = \nabla_{\dot{\mathbf{x}}}L = \mathbf{M}\dot{\mathbf{x}}$, and the Hamiltonian $H(\mathbf{x}, \mathbf{p_x}) = \dot{\mathbf{x}}^T\mathbf{p_x} - L(\mathbf{x}, \dot{\mathbf{x}})$, which has differential

$$\delta H = \mathbf{p_x}^T\delta\dot{\mathbf{x}} + \dot{\mathbf{x}}^T\delta\mathbf{p_x} - (\nabla_{\mathbf{x}}L)^T\delta\mathbf{x} - (\nabla_{\dot{\mathbf{x}}}L)^T\delta\dot{\mathbf{x}} = (\nabla_{\mathbf{x}}H)^T\delta\mathbf{x} + (\nabla_{\mathbf{p_x}}H)^T\delta\mathbf{p_x}. \tag{16}$$

The second equality holds because of the definition of δH. We can rewrite the equality as

$$\begin{bmatrix}\delta\mathbf{x}^T & \delta\mathbf{p_x}^T\end{bmatrix}\begin{bmatrix}\nabla_{\mathbf{x}}H + \nabla_{\mathbf{x}}L \\ \nabla_{\mathbf{p_x}}H - \dot{\mathbf{x}}\end{bmatrix} = \begin{bmatrix}\delta\mathbf{x}^T & \delta\mathbf{p_x}^T\end{bmatrix}\begin{bmatrix}\nabla_{\mathbf{x}}H + \dot{\mathbf{p}}_{\mathbf{x}} \\ \nabla_{\mathbf{p_x}}H - \dot{\mathbf{x}}\end{bmatrix} = 0. \tag{17}$$

The first equality holds by substituting in D'Alembert's principle (10). If all the coordinates are independent, we recover the Hamiltonian dynamics with implicit constraint (6). The Cartesian coordinates are not independent so we need to explicitly add constraints with Lagrangian multipliers to derive equations of motion. Assume we have $K = D - M$ holonomic constraints collected into a vector equality $\Phi(\mathbf{x}) = \mathbf{0}$, we can differentiate these constraints $\dot{\Phi}(\mathbf{x}) = (D_{\mathbf{x}}\Phi)\dot{\mathbf{x}} =$

$(D_{\mathbf{x}}\Phi)\mathbf{M}^{-1}\mathbf{p_x} = 0$ so that we form additional K constraints that depend on both $\mathbf{x}$ and $\mathbf{p_x}$. We denote $\mathbf{z} = (\mathbf{x}^T, \mathbf{p_x}^T)^T$ and the $2K$ constraints as $\Psi(\mathbf{z}) = (\Phi^T, \dot{\Phi}^T)^T$. Then elements in the differential $\delta\mathbf{z}$ are related by $(D_{\mathbf{z}}\Psi)\delta\mathbf{z} = \mathbf{0}$. We introduce $2K$ Lagrangian multipliers $\lambda_H \in \mathbb{R}^{2K}$ and add the scalar $\lambda_H^T(D_{\mathbf{z}}\Psi)\delta\mathbf{z} = 0$ to (17). We have

$$\delta\mathbf{z}^T[\nabla_{\mathbf{z}}H + \mathbf{J}\dot{\mathbf{z}} + (D_{\mathbf{z}}\Psi)^T\lambda_H] = 0, \tag{18}$$

where $\mathbf{J}$ is a symplectic matrix $\mathbf{J} = [\mathbf{0}, \mathbf{I}_D; -\mathbf{I}_D, \mathbf{0}]$ and $\mathbf{I}_D$ is the $D \times D$ identity matrix. Even though elements in $\delta\mathbf{z}$ are not independent, we can choose λ_H such that the only solution to (18) is one where the coefficients of all δz_i are zeros. Thus we get $2D$ equations. To solve for λ_H as a function of $\mathbf{z}$, we leverage $\dot{\Psi} = (D_{\mathbf{z}}\Psi)\dot{\mathbf{z}} = \mathbf{0}$ to eliminate $\dot{\mathbf{z}}$ and get

$$\lambda_H = -[(D_{\mathbf{z}}\Psi)\mathbf{J}(D_{\mathbf{z}}\Psi)^T]^{-1}(D_{\mathbf{z}}\Psi)\mathbf{J}\nabla_{\mathbf{z}}H. \tag{19}$$

Finally we get the equations of motion as $2D$ first order differential equations

$$\dot{\mathbf{z}} = \mathbf{J}\nabla_{\mathbf{z}}H - \mathbf{J}(D_{\mathbf{z}}\Psi)^T[(D_{\mathbf{z}}\Psi)\mathbf{J}(D_{\mathbf{z}}\Psi)^T]^{-1}(D_{\mathbf{z}}\Psi)\mathbf{J}\nabla_{\mathbf{z}}H. \tag{20}$$

2.5. Learning of Dynamics

All the dynamics derived above are first-order or second-order ordinary differential equations (ODE). The second-order ODEs, e.g., (15), can be turned into a set of first-order ODEs by choosing state $\mathbf{s}$, e.g., $\mathbf{s} = (\mathbf{x}^T, \dot{\mathbf{x}}^T)^T$. If we know how to learn first-order ODEs from data, we will be able to learn Lagrangian or Hamiltonian dynamics of any form above.

Neural ODE Chen et al. (2018) is a framework for learning a general first-order ODE $\dot{\mathbf{s}} = \mathbf{f}(\mathbf{s})$ with an unknown function $\mathbf{f}(\mathbf{s})$. The idea is to parametrize $\mathbf{f}(\mathbf{s})$ by a neural network. With an initial condition $\mathbf{s}_0$, we can use an ODE solver to solve the initial value problem for future states,

$$\hat{\mathbf{s}}_1, ..., \hat{\mathbf{s}}_T = \text{ODESolve}(\mathbf{f}_\theta(\mathbf{s}), \mathbf{s}_0, 1, ..., T), \tag{21}$$

where $\mathbf{f}_\theta(\mathbf{s})$ is the parametrization of $\mathbf{f}(\mathbf{s})$. We can then minimize the difference between the true states and the predicted states. As all the operations in the ODE Solver are differentiable, we can use back propagation to solve the optimization problem. To incorporate physics priors into this learning framework, we can parametrize unknown physical quantities in the derived dynamics above, e.g., mass, potential energy, the Lagrangian and the Hamiltonian. The difference between various recent models lies in what form of dynamics they use and what physical quantities they parametrize, which is explained in the next section.

3. Models Studied in this Survey

Since the Lagrangian and Hamiltonian dynamics we consider here conserve energy, the neural network models that exploit Lagrangian and Hamiltonian dynamics automatically enforce energy conservation. In this section, we briefly summarize recent energy-conserving neural network models implemented in this benchmark study. As the original names of these models might not reflect the difference between them, we renamed a few models to emphasize the difference, while citing the original papers that introduced them.

HNN: Greydanus et al. (2019) introduces Hamiltonian Neural Networks (HNN) to learn Hamiltonian dynamics (6) from data by parametrizing the Hamiltonian H using a neural network. The original implementation assumes the data are in the form of $(\mathbf{q}, \mathbf{p}, \dot{\mathbf{q}}, \dot{\mathbf{p}})_{0,\ldots,T}$. The learning is performed by feeding data $(\mathbf{q}, \mathbf{p})$ into the Hamiltonian neural network to get the scalar $H(\mathbf{q}, \mathbf{p})$, getting the right hand side (RHS) of (6) by automatic differentiation, and minimizing the violation of the dynamics (6) with data $(\dot{\mathbf{q}}, \dot{\mathbf{p}})$. In this benchmark, we assume we can only access position and velocity data. Thus, in our implementation, we additionally parametrize the mass matrix $\mathbf{M}(\mathbf{q})$ using a neural network and leverage Neural ODE to learn the dynamics.

HNN-structure: For rigid body systems, instead of parametrizing the Hamiltonian, we can leverage the structure of the Hamiltonian (8) and parametrize the mass matrix $\mathbf{M}(\mathbf{q})$ and potential energy $V(\mathbf{q})$ using neural networks. This allows us to learn Hamiltonian dynamics in the form of (9). This learning scheme is introduced as Symplectic ODE-Net (SymODEN) (Zhong et al., 2020a), which can jointly learn an additional control term. Dissipative SymODEN (Zhong et al., 2020b) jointly learns an additional dissipation term. Symplectic Recurrent Neural Network (SRNN) (Chen et al., 2020) also exploits the structure of the Hamiltonian by assuming a separable Hamiltonian.

CHNN: By explicitly enforcing constraints, Hamiltonian dynamics in the form of (20) can be learned from Cartesian coordinate data. Since the mass matrix under Cartesian coordinates is constant and sparse, elements in the mass matrix can be set as learnable parameters. Moreover, for a system of multiple rigid bodies, the mass matrix is block diagonal, which further reduces the number of learnable parameters (Finzi et al., 2020). As the inverse of mass matrix appears all the forms of Hamiltonian dynamics, the block diagonality and sparsity of the constant mass matrix under Cartesian coordinates facilitates the computation of matrix inverse. This learning scheme is introduced as Constrained HNN (CHNN) in Finzi et al. (2020).

LNN: Cranmer et al. (2020) introduces Lagrangian Neural Networks (LNN) to learn Lagrangian dynamics (3) from position and velocity data by parametrizing the Lagrangian L using a neural network. The benefit of this learning scheme is that it can learn any arbitrary Lagrangian, which does not need to be quadratic in velocity as in rigid body systems.

LNN-structure: For rigid body systems, the structure of the Lagrangian (4) can be leveraged and the mass matrix $\mathbf{M}(\mathbf{q})$ and potential energy $V(\mathbf{q})$ can be parametrized using neural networks to learn Lagrangian dynamics in the form of (5). This learning scheme is first introduced as Deep Lagrangian Network (DeLaN) in Lutter et al. (2019b) which jointly learn an additional control term for the purpose of controlling robotic systems. DeLaN is also used in Lutter et al. (2019a) to learn and control under-actuated systems. DeLaN assumes that position $\mathbf{q}$, velocity $\dot{\mathbf{q}}$ and acceleration $\ddot{\mathbf{q}}$ data can be accessed from sensor and the violation of (5) is minimized. For the purpose of benchmarking, we leverage Neural ODE to learn (5) without acceleration data.

CLNN: Learning Lagrangian dynamics with explicit constraints in the form of (15) from Cartesian coordinate data is introduced as Constrained LNN (CLNN) in Finzi et al. (2020). Similar to CHNN, the mass matrix properties of block diagonality and sparsity in each block are exploited to facilitate learning.

HNN-angle, HNN-structure-angle, LNN-angle, LNN-structure-angle: When enforcing constraints implicitly, we will often encounter angular coordinates. If we treat each angular coordinate as a variable in $\mathbb{R}^1$, the learning algorithm would not be able to infer, for example, 0 and

2π represents the same angle. In fact, each angular coordinate lies on the manifold $\mathbb{S}^1$ (circle) instead of manifold $\mathbb{R}^1$ (line). This problem can be solved by embedding an angular coordinate q as $(\cos q, \sin q) \in \mathbb{S}^1$. Thus, for the four models that enforce constraints implicitly (HNN, HNN-strucure, LNN, LNN-structure), we also implement their angle-aware version (names with "angle" suffix) by embedding each angular coordinate into $\mathbb{S}^1$ before feeding it into the neural networks. This angle-aware design is also implemented and studied in SymODEN (Zhong et al., 2020a).

For models above that parametrize the mass matrix $\mathbf{M}(\mathbf{q})$ using a neural network, we leverage the fact that for a real physical system, the mass matrix is positive definite and can be expressed as $\mathbf{M}(\mathbf{q}) = \mathbf{L}(\mathbf{q})\mathbf{L}(\mathbf{q})^T$ (Cholesky decomposition), where $\mathbf{L}(\mathbf{q})$ is a lower triangular matrix. Thus, instead of parametrizing the mass matrix or the inverse of mass matrix, we parametrize $\mathbf{L}(\mathbf{q})$, as done in Lutter et al. (2019b) and Zhong et al. (2020a)

4. Experiments

4.1. Experimental Setup

Figure 1 shows an N-pendulum system and a gyroscope. The N-pendulum is a set of 2D systems. The complexity of the dynamics increases as the number of pendulums increases. Except for the 1-pendulum, the systems are chaotic system which

Figure 1: **Left:** 4-pendulum; **Right:** gyroscope

means accurate prediction is not possible when $N \geq 2$ (Kwok et al., 2020). The gyroscope is a 3D system that exhibits complex dynamics including precession and nutation (Butikov, 2006). In this work, we simulate the 1-, 2- and 4-pendulum systems and the gyroscope to evaluate models.
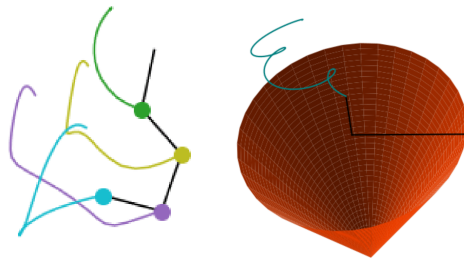
Data Generation and Training   For each of the 4 physical systems, we randomly generate 800 initial conditions of states and roll out simualtion trajectories for 100 time steps for training. We randomly generate another set of 100 initial conditions of 100 time steps for testing. Random Gaussian noises of standard deviation 0.01 are added independently to every coordinate and velocity. For the N-Pendulum systems, we use data in 2D Cartesian coordinates to train CHNN and CLNN, and data in planar angular coordinates to train the rest of the models. For the gyroscope, we use data in 3D Cartesian coordinates of 4 chosen points in the gyroscope as done in (Finzi et al., 2020) to train CHNN and CLNN, and data in Euler angles to train the rest of the models. We generate trajectories using RK4 integrator instead of symplectic integrators as done in Zhu et al. (2020), Sanchez-Gonzalez et al. (2019), Chen et al. (2020), Jin et al. (2020), Tong et al. (2020) because using RK4 integrator allows us to incorporate control and dissipation into the framework, whereas symplectic integrators does not. For the N-pendulum systems and the gyroscope system, the integration interval are 0.03s and 0.02s, respectively. As function approximators, we use fully-connected feedforward network with 3 hidden layers with 256 neurons/layers and Softplus activation. For learnable mass/moments in CLNN/CHNN, we implement them as learnable parameters passed through an exponential function, since the mass/moments are always positive for real physical systems. All models are trained with *AdamW* optimizer Loshchilov and Hutter (2019) for 1000 epochs and a learning rate of 0.001.

Metrics   We use $L(\mathbf{s}_{(0,...T)}, \hat{\mathbf{s}}_{(0,...,T)}) = \sum_{t=0}^{T} ||\hat{\mathbf{s}}_t - \mathbf{s}_t||_1$ as the loss function for training, since it has been shown that the robustness of L_1 loss to outliers benefits learning complex dynamics (Finzi
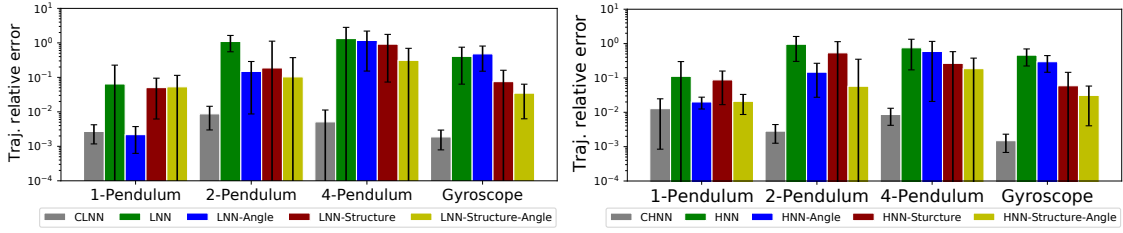
Figure 2: The trajectory relative error (log scale) in the states ($T = 4$). Each error is averaged over trajectories with 100 test initial conditions. The absolute error bars show the standard deviation in *log* scale. **Left**: Lagrangian models; **Right**: Hamiltonian models.
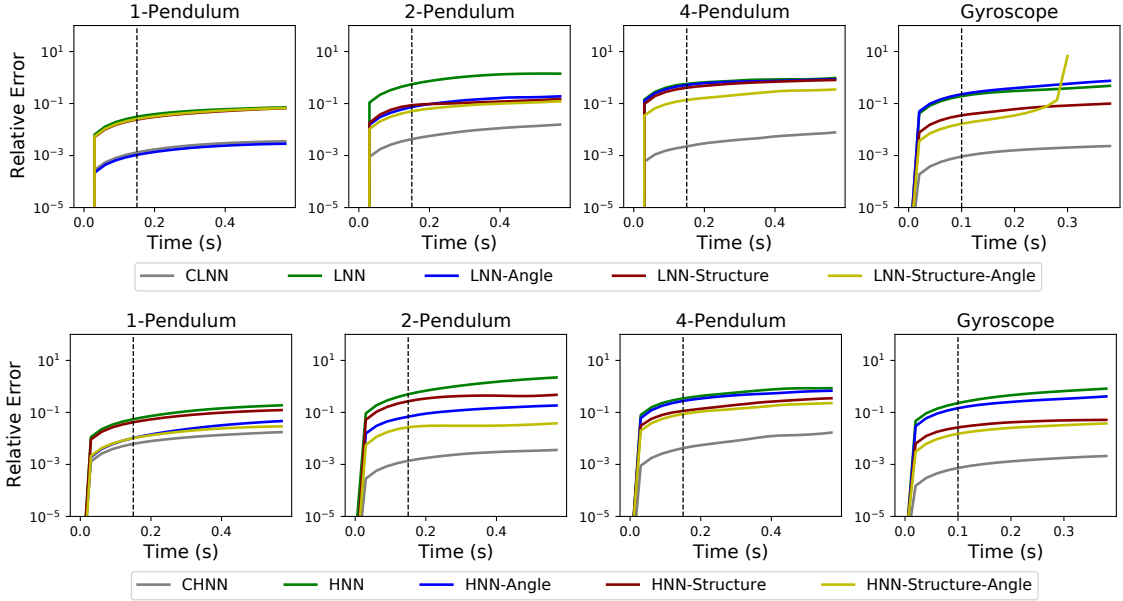

Figure 3: The relative error in the states (log scale). Each curve is averaged over the prediction trajectories of 100 test initial conditions. The vertical dashed lines shows the trajectory length in training. **Top**: Lagrangian models; **bottom**: Hamiltonian models.

et al., 2020). While training, we set $T = 4$ and select random snapshots of length 5 from each long trajectory. We evaluate the performance of each model by the *relative error* at each time step $||\hat{\mathbf{s}}_t - \mathbf{s}_t||_2/||\mathbf{s}_t||_2$ and the *trajectory relative error* $\sum_{t=0}^{T} ||\hat{\mathbf{s}}_t - \mathbf{s}_t||_2/||\mathbf{s}_t||_2$, which sums up relative errors along a trajectory. We also analyze each model's ability in conserving true energy with the *absolute error* in true energy $||\hat{E}_t - E_t||_2$. Since the absolute value of energy is not meaningful (potential energy is relative to an arbitrarily chosen zero position), we do not choose the relative error in true energy as the metric, which scales the absolute error by the absolute value of energy.

4.2. Results

Test error Figure 2 shows the trajectory relative error of 10 models on test trajectory data. The models with explicit constraints (CLNN and CHNN) consistently outperforms all the other models across all systems. The gap is the largest in the 3D system (Gyroscope) since with explicit constraints, the dynamics under Cartesian coordinates do not suffer from gimbal lock, which occurs in
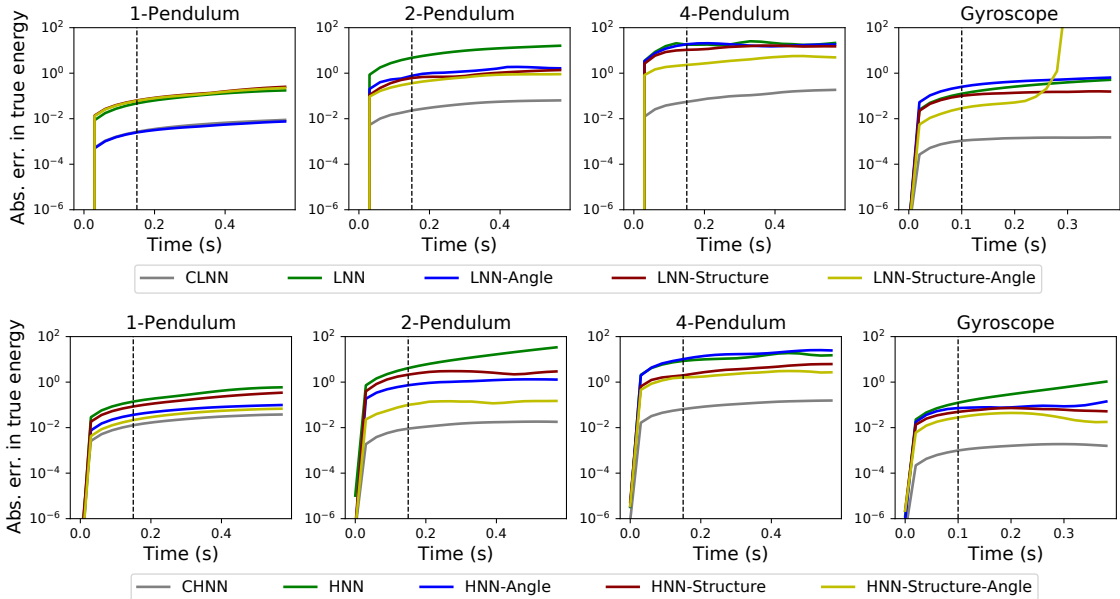
Figure 4: Absolute error in true energy (log scale). Each curve is averaged over the prediction trajectories of 100 test initial conditions. The vertical dash lines shows the trajectory length in training. **Top**: Lagrangian models; **bottom**: Hamiltonian models.

the dynamics under Euler angles (implicit constraints). For the models with implicit constraints, we can observe the benefit of leveraging a structured Lagrangian/Hamiltonian and angle-aware design. Comparing the angle-aware models with their non-angle-aware variants (e.g., LNN-Angle and LNN), we conclude that the angle-aware design in general benefits learning. This conclusion is consistent with our intuition since respecting the geometry of coordinates lets models identify q and $q \pm 2\pi$, which helps models learn dynamics. Comparing models that leverage structured Lagrangian and those that do not (e.g., LNN-Structure and LNN), we observe that the structure benefits the learning of 2 and 4-Pendulum systems while does not help the learning of 1-Pendulum and Gyroscope. The 1-Pendulum has simple dynamics, so it can be well learned without the structure. As the number of pendulums increases, the structure in Lagrangian/Hamiltonian helps recover dynamics that explain the data. For 3D Gyroscope, it is unclear why the structure does not help. We leave it as an open question for future work.

Long-term prediction. We evaluate 10 models on trajectories longer than those in training ($T = 4$). Figure 3 shows the average relative error over 100 test trajectories of length $T = 20$. The vertical dashed line shows the length of training trajectories. The 2-Pendulum and 4-Pendulum are chaotic systems so accurate long-term prediction is not possible. In general, CHNN and CLNN performs the best across all systems considered. We have similar observations on the role of angle-aware design and the structured Lagrangian/Hamiltonian.

Energy conservation All 10 models naturally incorporate the prior of energy conservation, which means the learned energy is conserved. However, the learned energy might not accurately represent the true energy. Figure 4 shows how the true energy calculated from prediction trajectories differ from the constant true energy along ground truth trajectories. We observe that although each model conserves its own learned energy, they all drift away from the constant true energy as time goes by.
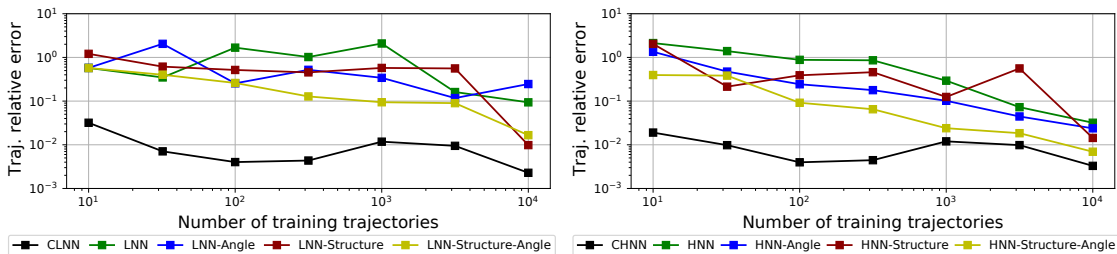
Figure 5: Trajectory relative error in the states (log scale) of 2-Pendulum with various sizes of training trajectories. Each curve is averaged over 100 test trajectories ($T = 4$). **Left**: Lagrangian models; **Right**: Hamiltonian models.

CLNN and CHNN has relative low drifts as compared to other models, which explains why they perform better in prediction.

Data Efficiency To compare the data efficiency of 10 models, we change the size of training dataset on the 2-Pendulum system from as low as 10 trajectories to as high as 10,000 trajectories, as shown in Figure 5. We observe that CLNN and CHNN outperforms all the other models. We also find that prior knowledge (structure/angle-aware) benefits data efficiency.

5. Conclusion and Discussion

In this work, we benchmark recent energy-conserving neural network models that rely on Lagrangian/Hamiltonian dynamics. CLNN and CHNN, which enforce constraints explicitly, have a significant advantage in prediction over models that enforce constraints implicitly as observed in Finzi et al. (2020), because the Lagrangian and Hamiltonian assume a much simpler form in Cartesian coordinates. This is due to the fact that the mass matrix under Cartesian coordinates is coordinate-independent, whereas the mass matrix depends on the coordinates in a complicated way when we use angular coordinates to describe the system. Moreover, for those models that need to learn a coordinate-dependent mass matrix (all Hamiltonian models except CHNN, and LNN-Structure, LNN-Structure-Angle), the initialized parametrization of the mass matrix tend to be ill-conditioned, which makes training unstable. In practice, a small positive constant is added to the diagonal elements of the mass matrix parametrization to alleviate this problem; however, this caveat complicates the training of these implicit models. For LNN and LNN-Angle, which do not learn the mass matrix, it is implicitly derived from the Hessian of the Lagrangian; since computing Hessian involves two gradient operations, these two models incur an additional computational cost and result in slower training. The advantage of models that enforce constraints implicitly lies in their easy implementation, whereas implementing CHNN and CLNN requires manually adding constraints for the systems of interest.

We also emphasize the similarity and difference between the analytical forms that the models learn and the intuition behind them. We conclude that angle-aware design in general benefits learning and imposing structures on the Lagrangian and the Hamiltonian benefits learning for 2D systems. In fact, CLNN and CHNN also leverage the structure in Lagrangian and Hamiltonian to simplify the dynamics. As the structure can be leveraged to design energy-based controllers (Zhong et al., 2020a), we believe it is promising to design controllers based on CLNN and CHNN.

Acknowledgement

References

Christine Allen-Blanchette, Sushant Veer, Anirudha Majumdar, and Naomi Ehrich Leonard. Lagnetvip: A lagrangian neural network for video prediction. *arXiv preprint arXiv:2010.12932*, 2020.

Eugene Butikov. Precession and nutation of a gyroscope. *European Journal of Physics*, 27(5):1071, 2006.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in neural information processing systems*, pages 6571–6583, 2018.

Zhengdao Chen, Jianyu Zhang, Martin Arjovsky, and Léon Bottou. Symplectic recurrent neural networks. In *International Conference on Learning Representations*, 2020.

Miles Cranmer, Sam Greydanus, Stephan Hoyer, Peter Battaglia, David Spergel, and Shirley Ho. Lagrangian neural networks. In *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*, 2020.

Ghanashyam Date. Lectures on constrained systems. *arXiv preprint arXiv:1010.2062*, 2010.

Marc Finzi, Ke Alexander Wang, and Andrew Gordon Wilson. Simplifying hamiltonian and lagrangian neural networks via explicit constraints. *Advances in Neural Information Processing Systems*, 33, 2020.

Herbert Goldstein, Charles Poole, and John Safko. Classical mechanics, 2002.

Ian Goodfellow, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT Press, 2016.

Samuel Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian neural networks. In *Advances in Neural Information Processing Systems*, pages 15379–15389, 2019.

David Haussler. Quantifying inductive bias: AI learning algorithms and Valiant's learning framework. *Artificial Intelligence*, 36(2):177–221, 1988.

Pengzhan Jin, Zhen Zhang, Aiqing Zhu, Yifa Tang, and George Em Karniadakis. Sympnets: Intrinsic structure-preserving symplectic networks for identifying hamiltonian systems. *Neural Networks*, 132:166–179, 2020.

Thomas M Kwok, Zheng Li, Ruxu Du, and Guanrong Chen. Modeling and experimental validation of the chaotic behavior of a robot whip. *Journal of Mechanics*, 36(3):373–394, 2020.

Shiyu Liang and Rayadurgam Srikant. Why deep neural networks for function approximation? In *International Conference on Learning Representations*, 2017.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

Michael Lutter, Kim Listmann, and Jan Peters. Deep lagrangian networks for end-to-end learning of energy-based control for under-actuated systems. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7718–7725. IEEE, 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.

Allan Pinkus. Approximation theory of the mlp model in neural networks. *Acta numerica*, 8(1): 143–195, 1999.

Manuel A. Roehrl, Thomas A. Runkler, Veronika Brandtstetter, Michel Tokic, and Stefan Obermayer. Modeling system dynamics with physics-informed neural networks based on lagrangian mechanics. In *21st IFAC World Congress*, 2020.

Alvaro Sanchez-Gonzalez, Victor Bapst, Kyle Cranmer, and Peter Battaglia. Hamiltonian graph networks with ode integrators. *arXiv preprint arXiv:1909.12790*, 2019.

Yunjin Tong, Shiying Xiong, Xingzhe He, Guanghan Pan, and Bo Zhu. Symplectic neural networks in taylor series form for hamiltonian systems. *arXiv preprint arXiv:2005.04986*, 2020.

Yaofeng Desmond Zhong and Naomi Leonard. Unsupervised learning of lagrangian dynamics from images for prediction and control. *Advances in Neural Information Processing Systems*, 33, 2020.

Yaofeng Desmond Zhong, Biswadip Dey, and Amit Chakraborty. Symplectic ode-net: Learning Hamiltonian dynamics with control. In *International Conference on Learning Representations*, 2020a.

Yaofeng Desmond Zhong, Biswadip Dey, and Amit Chakraborty. Dissipative symODEN: Encoding Hamiltonian dynamics with dissipation and control into deep learning. In *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*, 2020b.

Aiqing Zhu, Pengzhan Jin, and Yifa Tang. Deep hamiltonian networks based on symplectic integrators. *arXiv preprint arXiv:2004.13830*, 2020.