Physics-enhanced Neural Networks in the Small Data Regime

Jonas Eichelsdörfer[*], **Sebastian Kaltenbach**[*], **Phaedon-Stelios Koutsourelakis**
Professorship of Continuum Mechanics
Technical University of Munich
Munich, Germany
{jonas.eichelsdoerfer,sebastian.kaltenbach,p.s.koutsourelakis}@tum.de

Abstract

Identifying the dynamics of physical systems requires a machine learning model that can assimilate observational data, but also incorporate the laws of physics. Neural Networks based on physical principles such as the Hamiltonian or Lagrangian NNs have recently shown promising results in generating extrapolative predictions and accurately representing the system's dynamics. We show that by additionally considering the actual energy level as a regularization term during training and thus using physical information as inductive bias, the results can be further improved. Especially in the case where only small amounts of data are available, these improvements can significantly enhance the predictive capability. We apply the proposed regularization term to a Hamiltonian Neural Network (HNN) and Constrained Hamiltonian Neural Network (CHHN) for a single and double pendulum, generate predictions under unseen initial conditions and report significant gains in predictive accuracy.

1 Introduction

The present paper is concerned with the discovery of the dynamics of physical systems in the Small Data regime. Modern machine learning techniques without any inductive bias often fail in this task because the networks do not generalize well and lack robustness. To overcome these problems, we therefore combine latest advances in physics-enhanced Neural Networks based on dynamic invariants such as the Hamiltonian [1, 2, 3] or Lagrangian [4, 5] NNs with additional inductive bias to gain better predictive capabilities and to reduce the requisite training data. We propose a novel regularization term that can be readily added to the training loss function of most machine learning frameworks and show that it leads to predictive performance and data efficiency that outperforms the original framework for all tested examples and under initial conditions not contained in the training data.

This work shares similarities with approaches that use physical laws as regularization terms or augment the loss function as in [6, 7, 8, 9]. Another line of work closely related to ours pertains to the use of inductive bias with regards to the long-term stability [10] or embedded symmetries [11] of physical systems. Moreover, Graph Neural Networks [12, 13] can be tailored to physical settings by e.g. combining them with the aforementioned Hamiltonian [14].

[*]equal contribution

2 Physics-enhanced Neural Networks

Domain knowledge originating from physics can be used in various ways to enhance the performance of neural networks. This section introduces physics-enhanced neural networks based on dynamic invariants and a novel regularization term based on the energy level/difference of the training data.

2.1 Hamiltonian Neural Networks

The basic idea of HNNs is to incorporate the principle of Hamiltonian mechanics as inductive bias. Instead of approximating the governing equations for each degree of freedom, the Hamiltonian equations are invoked, i.e. $\dot{\boldsymbol{q}} = \frac{\partial H}{\partial \boldsymbol{p}}$, $\dot{\boldsymbol{p}} = \frac{\partial H}{\partial \boldsymbol{q}}$, and solely a differentiable representation of the system's Hamiltonian H is learned. We note that for isolated systems, the Hamiltonian is written in terms of generalized coordinates $\boldsymbol{q}$ and momenta $\boldsymbol{p}$, and it expresses the system's total energy which is conserved during the dynamic evolution.
Greydanus [1] proposed to learn the Hamiltonian on the basis of N observations of the system's state variables $\boldsymbol{z}^{(i)} = (\boldsymbol{q}^{(i)}, \boldsymbol{p}^{(i)})$ as well as their time derivatives $\dot{\boldsymbol{z}}^{(i)} = (\dot{\boldsymbol{q}}^{(i)}, \dot{\boldsymbol{p}}^{(i)}), i = 1, \ldots, N$ with the following loss function $\mathcal{L}$:

$$\mathcal{L}(\boldsymbol{\phi}) = \frac{1}{N} \sum_{i=1}^{N} \left\| \frac{\partial H_{\boldsymbol{\phi}}(\boldsymbol{z}^{(i)})}{\partial \boldsymbol{p}} - \dot{\boldsymbol{q}}^{(i)} \right\|^2 + \left\| \frac{\partial H_{\boldsymbol{\phi}}(\boldsymbol{z}^{(i)})}{\partial \boldsymbol{q}} + \dot{\boldsymbol{p}}^{(i)} \right\|^2 \tag{1}$$

where $\boldsymbol{\phi}$ are the tunabe parameters (e.g. NN weights/biases). Since the Hamiltonian dynamics automatically respect the conservation of energy, a significant improvement in long term simulation accuracy can be achieved. As the loss function is not based on the absolute value of H value but its gradient, the learned Hamiltonian may differ from the system's total energy by a constant factor.
We note that the accuracy in the learned H must be extremely high in order to provide good estimates of its derivatives which are needed for predictive purposes and which becomes increasingly challenging as the dimension of $\boldsymbol{z}$ increases.

2.2 Constrained HNNs

Another promising approach to represent the Hamiltonian of a physical system is the constrained Hamiltonian method. Finzi et al. [3] recognized that frequently the structure of the Hamiltonian is significantly simpler if expressed in Cartesian rather than generalized coordinates and achieved better and more data-efficient results than the HNN method in multibody dynamics. The algorithm proposed employs a mapping from the Cartesian coordinates onto the constraint manifold described by the generalized coordinates. The downside of the method is that in order to use this the mapping, a linear system of equations must be solved at every inference step, both during the network training and prediction.

2.3 Lagrangian Neural Networks

To overcome the restriction of employing generalized coordinates in the Hamiltonian formalism, Cranmer et al.[5] proposed Lagrangian Neural Networks (LNNs). Based on the Lagrangian the acceleration of any degree of freedom can readily be calculated and a loss function can be stated in terms of the discrepancy between the true and the model-predicted acceleration. This method therefore is more generally applicable than the Hamiltonian formalism, but does involve a matrix inversion during each training step [5].

2.4 Proposed regularization

In this work, the loss function of physics-enhanced neural networks such as HNN and CHNN is extended by an additional, physics-informed, regularization term which penalizes the difference between the learned Hamiltonian $H_{\boldsymbol{\phi}}(\boldsymbol{z})$ and target values of the system's total energy level $\hat{H}$, thereby setting the level of the predicted Hamiltonian. It is sufficient to compute $\hat{H}$ once for every initial condition contained in the training data. The proposed regularization term restricts the inferred solution to the numerical value of the total energy data. The augmented loss function is given by

$$\bar{\mathcal{L}}(\boldsymbol{\phi}) = \frac{1}{N}\sum_{i=1}^{N}\left\|\frac{\partial H_{\boldsymbol{\phi}}(\boldsymbol{z}^{(i)})}{\partial \boldsymbol{p}} - \dot{\boldsymbol{q}}^{(i)}\right\|^2 + \left\|\frac{\partial H_{\boldsymbol{\phi}}(\boldsymbol{z}^{(i)})}{\partial \boldsymbol{q}} + \dot{\boldsymbol{p}}^{(i)}\right\|^2 + \lambda_H (H_{\boldsymbol{\phi}}(\boldsymbol{z}^{(i)}) - \hat{H}_i)^2 \ , \quad (2)$$

The total energy of the system must therefore be collected/computed as additional training data. We note that the zero level of this total energy can be set arbitrarily. As during training and for predictions only the gradients of the Hamiltonian are relevant, the actual zero level does not matter. From a physical point of view, we are passing information regarding the energy level difference between the different training data points to the algorithm. This regularization becomes increasingly important in the small-data regime. For real world problems, the generation of large amounts of training data is often prohibitively expensive. It is thus crucial to improve the robustness of the algorithms under limited training data. If data on the total energy of a system is available, it can thus be leveraged to improve the long term accuracy and generalization capability of the learned Hamiltonian dynamics as we demonstrated in the sequel.
The proposed regularization involves an additional hyperparameter λ_H. The optimal value of λ_H is dependent on the problem at hand and has to be found by cross-validation.

3 Numerical Illustrations

The proposed regularization term is applied to HNNs and CHNNs for a single and a double pendulum. All numerical schemes discussed in this work are implemented with the help of the Jax framework [15], a library for high-performance machine learning research based on the well-known Autograd [16] and TensorFlow [17] projects. Training data was generated by numerical integration of Eq. (3), Eq. (4) with a 4^{th}-order Runge-Kutta scheme and a time-step of $0.1s$. The results of a black-box MLP model [1, 3] serve as baseline in Tables 1, 2. The code can be found at `https://github.com/pkmtum/Physics-enhanced_NN_SmallData`.

3.1 Single Pendulum

The first example considered is the single-degree-of-freedom pendulum which involves a point mass m, suspended from its pivot with a massless rod of length l, is swinging frictionlessly. When displaced, gravity causes the pendulum to oscillate periodically about the equilibrium position. The angle θ between the pendulum rod and the vertical axis is chosen as the sole generalized coordinate. The equation of motion reads

$$\ddot{\theta} + \frac{g}{l}\sin\theta = 0. \quad (3)$$

The mass m and l are set to unity while the acceleration of gravity is set to $g = 9.81$. The conjugate momentum is $p = ml\dot{\theta}$. The pendulum is simulated forward for 150s starting from four different initial angles, equally spaced between 0 and 180 degrees and with 0 initial velocity. For our experiments we considered two training sets: a full dataset (f) that comprises a total of 600 data points, i.e. one data point per second, and a smaller dataset (s) with a total of only 64 data points, i.e. a data point every 10 seconds. For LNNs, the Lagrangian is expressed in terms of the angular coordinates $(\theta, \dot{\theta})$. For HNNs, the Hamiltonian is expressed with respect to (θ, p) and the CHNNs in terms of the Cartesian coordinates (x, y, p_x, p_y).

The Hamiltonian/Lagrangian is parametrized with a MLP with two hidden layers, with 32 hidden units each and a softplus activation function. The network is trained on full batch data for 150000 epochs using the Adam [18] optimization scheme. The learning rate is set to decay in two steps from $1e-2$ to $1e-3$ after the first 50000 epochs and finally to $1e-4$ after another 50000 epochs.

For testing purposes, we simulated the system for $100s$ starting from ten random initial conditions that were **not contained** in the training data. As error metrics we chose the absolute value of the energy error ΔE, for which we report the mean value and the standard deviation, as well as the maximum energy error $\max \Delta E$. The regularization parameter λ_H was chosen by cross-validation to $\lambda_H = 0.07$ for HNN and $\lambda_H = 0.01$ for CHHN.
The results are displayed in Table 1 where comparisons with LNNs are also reported. The physics-enhanced neural networks outperformed the baseline approach in all cases and the novel regularization

Table 1: Single pendulum error metrics for networks trained on the full dataset (f) and smaller dataset (s). All values are reported as percentages of the maximum potential energy.

Scheme	ΔE_f	$\max \Delta E_f$	ΔE_s	$\max \Delta E_s$
Baseline	0.3706 ± 0.3767	1.5115	2.4763 ± 2.1174	8.5754
HNN	0.0022 ± 0.0020	0.0107	0.0934 ± 0.1208	0.6008
HNN + H-Reg.	**0.0011 ± 0.0012**	**0.0058**	**0.0181 ± 0.0261**	**0.1760**
CHNN	0.0024 ± 0.0033	0.0257	0.2920 ± 0.4904	2.3802
CHNN + H-Reg.	**0.0019 ± 0.0022**	**0.0108**	**0.0584 ± 0.1244**	**1.1310**
LNN	0.0043 ± 0.0039	0.0132	0.2091 ± 0.2307	0.7806

Table 2: Double pendulum error metrics for networks trained on the full dataset (f) and the smaller dataset (s). All values are reported as percentages of the maximum potential energy.

Scheme	ΔE_f	$\max \Delta E_f$	ΔE_s	$\max \Delta E_s$
Baseline	14.201 ± 9.2108	48.267	-	-
HNN	0.809 ± 1.0147	4.962	18.393 ± 22.8308	186.648
HNN + H-Reg.	**0.353 ± 0.4175**	**2.320**	**6.086 ± 6.6419**	**33.000**
CHNN	0.052 ± 0.0507	0.250	0.344 ± 0.4350	3.625
CHNN + H-Reg.	**0.021 ± 0.0227**	**0.178**	**0.166 ± 0.2004**	**1.705**
LNN	1.290 ± 2.2827	12.168	-	-

term was able to significantly reduce both error metrics even further. The performance was superior for the HNN and the relative improvement was larger in the Small Data regime i.e. for the training data set (s).

3.2 Double Pendulum

The double pendulum is a chaotic nonlinear physical system which consists of two point mass pendulums that are attached to one another and subjected to a constant gravitational force. Each mass is attached to a rod and these are considered massless. The generalized coordinates describing the system's state are chosen to be the angles which the first and second rod form with the vertical direction. The geometrical parameters l_1, l_2, m_1, m_2 representing the rods' length and point masses are set to unity. The double pendulum's dynamics are described by a system of coupled differential equations. As for the single pendulum, a solution was obtained by numerical integration of

$$\begin{aligned} \ddot{\theta}_1 + \frac{l_2}{l_1}\frac{m_2}{m_1+m_2}\cos(\theta_1-\theta_2)\ddot{\theta}_2 &= -\frac{l_2}{l_1}\frac{m_2}{m_1+m_2}\dot{\theta}_2^2\sin(\theta_1-\theta_2) - \frac{g}{l_1}\sin\theta_1 \\ \ddot{\theta}_2 + \frac{l_1}{l_2}\cos(\theta_1-\theta_2)\ddot{\theta}_1 &= \frac{l_1}{l_2}\dot{\theta}_1^2\sin(\theta_1-\theta_2) - \frac{g}{l_2}\sin\theta_2 \quad . \end{aligned} \tag{4}$$

The MLP parametrization of the double pendulum's Hamiltonian/Lagrangian was chosen to consist of two hidden layers with 128 hidden units each and the softplus activation function. We trained the model on a full size dataset (f) with 6000 data points, i.e. ten data point per second, and small dataset (s) with 600 data points, i.e. one data point per second. The network was trained on full batch data for 150000 epochs using Adam [18] optimizer. The identical learning rate schedule as in the single pendulum case is used. During testing, we simulated the system for for $100s$ starting from ten random initial conditions that were **not contained** in the training data. The regularization parameter λ_H was chosen by cross-validation to $\lambda_H = 0.2$ for HNN and $\lambda_H = 0.005$ for CHHN.

The results are displayed in Table 2 where comparisons with LNNs are also reported. The physics-enhanced neural networks outperformed the baseline approach [2] in all cases and the novel regularization parameter was able to significantly reduce both error metrics even further. This improvement was most striking in the Small Data regime. We note that the CHNN algorithm performed much better than HNN for this example and the LNN diverged for the small data case.

[2] For the small training set the baseline approach diverged

4 Conclusions

Augmenting the loss function of physics-enhanced neural networks with an additional regularization term can lead to better predictive capabilities, especially in the Small Data regime. We showed that this approach significantly outperforms the current methods for the single and double pendulum example and under initial conditions not contained in the training data.
A current disadvantage is that the regularization parameter λ_H has to be chosen for each training data set separately. Here, further improvements based on advanced techniques regarding the training of PINNS [19, 20] are possible.

References

[1] Sam Greydanus, Misko Dzamba, and Jason Yosinski. "Hamiltonian neural networks". In: *arXiv preprint arXiv:1906.01563* (2019).

[2] Peter Toth et al. "Hamiltonian generative networks". In: *arXiv preprint arXiv:1909.13789* (2019).

[3] Marc Finzi et al. "Generalizing convolutional neural networks for equivariance to lie groups on arbitrary continuous data". In: *International Conference on Machine Learning*. 2020, pp. 3165–3176.

[4] Michael Lutter, Christian Ritter, and Jan Peters. "Deep lagrangian networks: Using physics as model prior for deep learning". In: *arXiv preprint arXiv:1907.04490* (2019).

[5] Miles Cranmer et al. "Lagrangian neural networks". In: *arXiv preprint arXiv:2003.04630* (2020).

[6] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations". In: *Journal of Computational Physics* 378 (2019), pp. 686–707.

[7] Bethany Lusch, J Nathan Kutz, and Steven L Brunton. "Deep learning for universal linear embeddings of nonlinear dynamics". In: *Nature communications* 9.1 (2018), pp. 1–10.

[8] Yinhao Zhu et al. "Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data". In: *Journal of Computational Physics* 394 (2019), pp. 56–81.

[9] Sebastian Kaltenbach and Phaedon-Stelios Koutsourelakis. "Incorporating physical constraints in a deep probabilistic machine learning framework for coarse-graining dynamical systems". In: *Journal of Computational Physics* 419 (2020).

[10] Sebastian Kaltenbach and Phaedon-Stelios Koutsourelakis. "Physics-aware, probabilistic model order reduction with guaranteed stability". In: *ICLR* (2021).

[11] Robin Walters, Jinxi Li, and Rose Yu. "Trajectory prediction using equivariant continuous convolution". In: *ICLR* (2021).

[12] Franco Scarselli et al. "The graph neural network model". In: *IEEE Transactions on Neural Networks* 20.1 (2008), pp. 61–80.

[13] Peter W. Battaglia et al. "Relational inductive biases, deep learning, and graph networks". In: *arXiv preprint arXiv:1806.01261* (2018).

[14] Alvaro Sanchez-Gonzalez et al. "Hamiltonian graph networks with ode integrators". In: *arXiv preprint arXiv:1909.12790* (2019).

[15] James Bradbury et al. *JAX: composable transformations of Python+NumPy programs*. 2018. URL: `http://github.com/google/jax`.

[16] Dougal Maclaurin, David Duvenaud, and Ryan P. Adams. "Autograd: Effortless gradients in numpy". In: *ICML 2015 AutoML Workshop*. Vol. 238. 2015, p. 5.

[17] Martin Abadi et al. *TensorFlow: Large-scale machine learning on heterogeneous systems*. 2015. URL: `https://www.tensorflow.org/`.

[18] Diederik P. Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[19] Sifan Wang, Yujun Teng, and Paris Perdikaris. "Understanding and mitigating gradient pathologies in physics-informed neural networks". In: *arXiv preprint arXiv:2001.04536* (2020).

[20] Sifan Wang, Xinling Yu, and Paris Perdikaris. "When and why PINNs fail to train: A neural tangent kernel perspective". In: *arXiv preprint arXiv:2007.14527* (2020).

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] see the claims in the Abstract and the results in the numerical illustrations section, where the new method is validated against current algorithms.

(b) Did you describe the limitations of your work? [Yes] Yes, in the conclusions the limitations are discussed.

(c) Did you discuss any potential negative societal impacts of your work? [N/A] anticipated impact pertains primarily to the scientific community. Use in real-life applications would require additional adjustments and expert input.

(d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes] The paper conforms to the guidelines.

2. If you are including theoretical results...

   (a) Did you state the full set of assumptions of all theoretical results? [N/A]

   (b) Did you include complete proofs of all theoretical results? [N/A]

3. If you ran experiments...

   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] all code and data is published on GitHub

   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] see the numerical illustrations section

   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We report the standard deviation of the energy error as well as the maximum energy error.

   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No] This is not mentioned in the paper, but due to working with small data all experiments could be run on a Lenovo ThinkPad L380 notebook with Intel i5-8250U CPU and 8 GB RAM.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

   (a) If your work uses existing assets, did you cite the creators? [Yes] Yes, existing approaches used were cited.

   (b) Did you mention the license of the assets? [N/A]

   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]

   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]