Learning Physics Constrained Dynamics Using Autoencoders

IZABELLA PAVLOVA, Technical University of Munich

With the increasing integration of neural networks (NN) in physics and control systems, this paper explores the application of NN models for estimating states and physical parameters from observations when the behavior of the system is described with dynamic equation provided. To address the problem we use the autoencoder with Latent Physics (ALPS) model, which consists of an encoder, estimator, physics simulator, and decoder. The encoder estimates states from observations, the estimator predicts physical parameters from these states, the physics simulator generates state trajectories consistent with the laws of physics and the decoder reconstructs observations from the simulated states. Furthermore, the paper demonstrates the benefits of incorporating self-attention mechanisms and Fourier mapping techniques, supported by the Neural Tangent Kernel (NTK) theory. Moreover, performance, limitations and directions for future research are discussed.

Additional Key Words and Phrases: Dynamic system parameters estimation, control systems, neural tangent theory, Fourier feature mapping

1 INTRODUCTION

We explore the problem of estimating states (e.g., position and velocity) and physical parameters (e.g., friction, elasticity) from a sequence of observations when provided a dynamic equation that describes the behavior of the system. Estimated states ensure the representation learned by neural networks to be informative and might be useful in some applications where we want to know the state of other objects in the system. For example, in case of self-driving car, we would like to predict the speed of other vehicles around using observations from cameras and adjust the car trajectory considering this parameters. Knowing the physical parameters of a system is fundamental for modeling, predicting, controlling, optimizing, diagnosing faults, and ensuring the safety and reliability of the system in various applications.

We consider the continuous-time system

$$\dot{x} = Ax + Bu; \qquad o = g(x), \tag{1}$$

where $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times p}$, and $x \in \mathbb{R}^n$, $u \in \mathbb{R}^p$, $o \in \mathbb{R}^q$, denote the state, input, and observation, respectively. The function g provides a partial observation o of x. In addition, we have partial knowledge of A and B from physics. We assume the system (1) to be linear. Moreover, in practice we can only make discrete observation — pixel images or direct measurements. For simplicity, we assume the times are equally spaced $t = 0, 1, \ldots$. At each sample time t, we have the window of observations $\{o_s\}_{s=t-\tau+1}$, where τ is the window length. We assume the sampling rate satisfies the Nyquist rate, meaning the sampling rate enables us to represent a continuous-time signal in digital form without loosing the information. In addition, to make the problem well-defined we assume that for a sufficiently large τ, the sequence of states and inputs $\{(x_s, u_s)\}_{s=t-\tau+1}$ uniquely specifies the unknown parameter θ.

This report is a part of the lecture, Master-Seminar — Deep Learning in Physics, Informatics 15, Technical University of Munich.
The original work is introduced by [Yang et al. 2022].

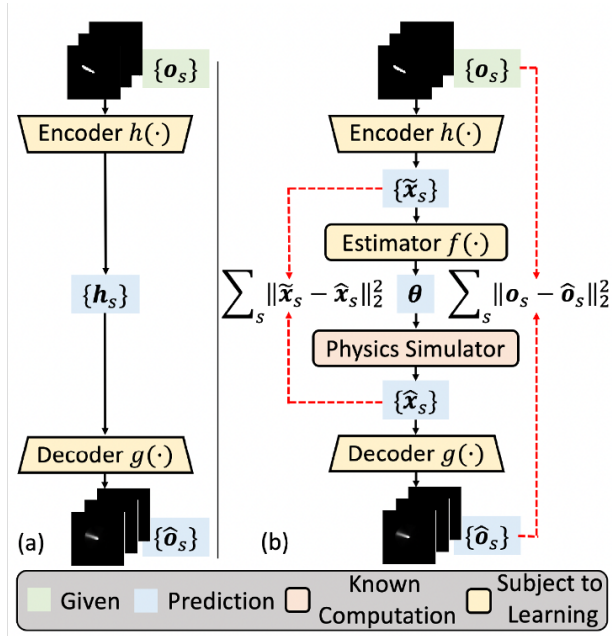

Fig. 1. ALPS Architecture

2 BACKGROUND

Physics-informed neural networks.
Koopman-inspired neural networks.

3 ALPS ARCHITECTURE

The ALPS model (Fig. 1) estimates states and physical parameters of the system using four main components. Firstly, encoder h estimates states $\tilde{x}_s$ from an observation sequence o_s. Secondly, estimator f predicts physical parameters θ from the states $\tilde{x}_s$. Thirdly, using the initial state and predicted parameters θ physics simulator generates a state trajectory $\hat{x}_s$ consistent with the laws of physics. Lastly, a decoder g reconstructs observations $\hat{o}_s$ from the state trajectory $\hat{x}_s$. The autoencoder is trained to minimize the observation reconstruction loss $\sum_s \|o_s - \hat{o}_s\|_2^2$. In addition, the encoder h is trained to minimize the sum of squared state errors $\sum_s \|\tilde{x}_s - \hat{x}_s\|_2^2$.

The encoder network. Depending on the type of observations — pixel images or direct measurements — a convolution or feed-forward network is used to represent an observation o as a vector embedding z'. To understand the dynamics within the data, the local and global context of the dynamics are extracted by aggregating the vector embedding z' using a self-attention network [Vaswani et al. 2017].

Firstly, to incorporate positional information, a positional encoding is added to the embeddings which are then stacked over τ steps to form a matrix. Secondly, to calculate the attention softmax is taken over the τ steps. Thirdly, to allow the model to jointly attend

to information from different representation subspaces at different positions the multi-head attention mechanism is applied (2,3).

$$\text{Multihead} = \text{Concat}(\text{head}_1, \ldots, \text{head}_h) \cdot W^O \tag{2}$$

$$\text{head}_i = \text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right) V, \tag{3}$$

where Q, K, V represent the query, key, and value matrices of dimensionality d respectively and W^O are learnable matrices.
Finally, using Gaussian and Mises distributions as posterior for translational and rotational coordinates accordingly, a feedforward network produces the parameters of the distribution for each state in the sequence.
The parameter estimator network. It was shown that multilayer perceptrons (MLPs) are affected by spectral bias [Rahaman et al. 2019] i.e. lower frequencies are learned first. Therefore, for systems that involve periodical or vibrational behavior, such as pendulum and oscillator, the input is passed through a Fourier transform on each component j of state trajectory (4) in order to capture high frequency content in the data [Tancik et al. 2020]. For other systems, which do not have periodic behavior the transformation is not needed. For each Fourier feature mapping a residual network [He et al. 2016] is used to get a representation. The parameter estimator MLP network predicts physical parameters θ by taking a concatenation of the magnitudes of Fourier features $|\tilde{X}_\omega(j)|$ from states $\tilde{x}_s$.

$$\tilde{X}_\omega(j) := \sum_{k=t-\tau+1}^{t} \tilde{x}_k(j) \left(\cos\left(\frac{2\pi}{\tau}\omega k\right) - i \cdot \sin\left(\frac{2\pi}{\tau}\omega k\right)\right) \tag{4}$$

The Neural Tangent Kernel theory. The choice of using magnitudes of the Fourier features for periodic systems can be justified by looking at neural tangent kernels [Jacot et al. 2018] of different Fourier mappings. Consider a set of labelled training data $\{(v_i, y_i)\}_{i=1}^m$ with $v_i \in \mathbb{R}^n$, $y_i \in \mathbb{R}$, and $i \in [1 : m]$, where state trajectory is $\mathbf{v} = [x_0, x_1, \ldots, x_{\tau-1}]^T$ and $\phi : \mathbb{R}^n \to \mathbb{R}^r$ with kernel $k(\mathbf{v}_i, \mathbf{v}_j) = (\mathbf{v}_i)^T(\mathbf{v}_j)$ is a feature map. Set $\mathbf{y} = [y_1, \ldots, y_m]^T \in \mathbb{R}^m$, $K = [k(\mathbf{v}_i, \mathbf{v}_j)] \in \mathbb{R}^{m \times m}$ denote the kernel matrix for the training examples and $k(\mathbf{v}) = [k(\mathbf{v}_i, \mathbf{v})] \in \mathbb{R}^m$ denote the vector of kernel evaluations $k(\mathbf{v}_i, \mathbf{v})$, $i \in [1, m]$, for a test sample $\mathbf{v} \in \mathbb{R}^n$. The resulting kernel regression predictor is $\hat{y}(\mathbf{v}) = \mathbf{y}^T K^{-1} k(\mathbf{v})$. According to the NTK theory when the number of neurons in each layer of fully-connected deep network with weights $\mathbf{w}$ initialized from a Gaussian distribution $\mathcal{N}$ tends to infinity, and the learning rate for stochastic gradient descent tends to zero, the neural network estimator $\hat{y}(\mathbf{v}; \mathbf{w})$ converges to the kernel regression solution.
The studied feature maps include the Fourier feature mapping, their magnitudes and their phases (5).

$$\begin{aligned} \phi_{\text{DFT}}(\mathbf{v}) &= [X_0, \ldots, X!, \ldots, X_{\tau-1}]^T \in \mathbb{R}^\tau \\ \phi_{\text{MAG}}(\mathbf{v}) &= [|X_0|, \ldots, |X_{\tau-1}|]^T \in \mathbb{R}^\tau \\ \phi_{\text{PHA}}(\mathbf{v}) &= [\arg(X_0), \ldots, \arg(X_{\tau-1})]^T \in \mathbb{R}^\tau \end{aligned} \tag{5}$$

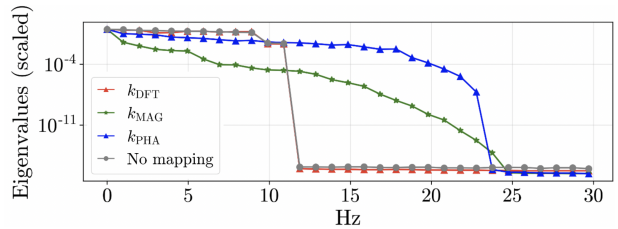

Fig. 2. Kernel spatial of the kernel matrices

Setting $C_k = [\cos\left(\frac{2\pi}{\tau}k_i 0\right) - \sin\left(\frac{2\pi}{\tau}k_j 0\right)] \in \mathbb{R}^{\tau \times \tau}$, the kernel functions of the mappings can be calculated the following way (6).

$$\begin{aligned} k_{\text{DFT}}(\mathbf{v}_1, \mathbf{v}_2) &= \frac{1}{\tau}\sum_{k=0}^{\tau-1} \mathbf{v}_1^T C_k \mathbf{v}_2; \\ k_{\text{MAG}}(\mathbf{v}_1, \mathbf{v}_2) &= \frac{1}{\tau}\sum_{k=0}^{\tau-1} \sqrt{\mathbf{v}_1^T C_k \mathbf{v}_1 \mathbf{v}_2^T C_k \mathbf{v}_2}; \\ k_{\text{PHA}}(\mathbf{v}_1, \mathbf{v}_2) &= \phi_{\text{PHA}}(\mathbf{v}_1)^T \phi_{\text{PHA}}(\mathbf{v}_2). \end{aligned} \tag{6}$$

In order to analyze the kernel spatial of different mapping we need to look at eigenvalues of corresponding kernel matrices of the composed NTK. Firstly, the kernel function is applied to every pair of samples from the trajectory $\mathbf{v}$. Secondly, the kernel regression is performed using the kernel matrix K to predict the output $\hat{y}(\mathbf{v})$. Lastly, the NTK is computed (7).

$$k_{\text{NTK}}(v_i, v_j) = \mathbb{E}_{w \sim \mathcal{N}}\left[\nabla_w \hat{y}^{(t)}(v_i; w)^\top \nabla_w \hat{y}^{(t)}(v_j; w)\right] \tag{7}$$

Spatial plot shows that k_{MAG} and k_{PHA} have a slower decay in the high-frequency domain and therefore k_{MAG} preserve most of the high-frequency components in the data (Fig. 2).
The physics simulator. The core of the physics simulator is a neural ordinary differential equation (ODE). Neural ODE [Chen et al. 2018] is a type of neural network (NN) models, where instead of specifying a discrete sequence of hidden layers, the derivative of the hidden state is parameterized using a NN. Given the initial state $\tilde{x}_s$ predicted by the encoder and physical parameters θ predicted by the estimator, the ODE solver rollouts state trajectories $\hat{x}_s$ (8).

$$\begin{aligned} &\{\hat{x}_s\}_{s=t-\tau+1}^{t} : \hat{x}_{t-\tau+1}, \ldots, \hat{x}_{t-1}, \hat{x}_t \\ &= \text{ODESolver}(\tilde{x}_{t-\tau+1}, \dot{x} = A(\theta)x + B(\theta)u, \tau, \triangle), \end{aligned} \tag{8}$$

where $\triangle$ is a sampling start interval.
The decoder network. Depending on the type of the input observations — images or sensor measurements — the decoder network is either a deconvolutional or a feedforward network, that generates a reconstructed observation $\hat{o}_s$ from each of the states $\hat{x}_s$, simulated by the ODE solver.
The loss function. The loss function consists of three loses (9). Firstly, the variational autoencoder (VAE) loss is used to train the encoder h, the estimator f and the decoder g. Secondly, the observation reconstruction loss used to minimize the error between input o_s and reconstructed $\hat{o}_s$ observations, which improves the image reconstruction quality. Thirdly, the state reconstruction loss encourages states, defined by encoder $\tilde{x}_s$ to follow the ones, generated by

	Pendulum			Mass-Spring-Damper			Two-body		
	SE	OE	PE	SE	OE	PE	SE	OE	PE
CDM	345.22	1766.70	–	0.99	**5.69×10^8**	–	50.23	**2.03×10^8**	–
Autoencoder	3041.12	**600.84**	–	7.63	7.42×10^8	–	95.80	2.71×10^8	–
ALPS	**86.48**	1696.91	**0.06**	**0.29**	7.43×10^8	**0.60×10^6**	**0.45**	2.59×10^8	**0.02**
w/o Fourier feat.	90.82	1773.39	0.29	0.93	7.44×10^8	2.28×10^6	1.86	2.56×10^8	**0.02**
w/o self-attention	181.22	1950.77	**0.06**	1.85	7.44×10^8	1.87×10^6	511.49	2.72×10^8	0.27

Fig. 3. Performance of tested networks in the visual tasks

physics simulator $\hat{x}_s$ and constrains the encoder from predicting arbitrary sequences.

$$L = \sum_{s=t-\tau+1}^{t} D_{KL}(Q(\tilde{x}_s|o_s)||P(\tilde{x}_s)) + \sum_{s=t-\tau+1}^{t} \|o_s - \hat{o}_s\|_2^2 + \sum_{s=t-\tau+1}^{t} \|\tilde{x}_s - \hat{x}_s\|_2^2 \tag{9}$$

4 PERFORMANCE EVALUATION

To evaluate the ALPS performance and an effect of chosen self-attention and Fourier mapping mechanisms the following baselines methods were selected for comparison.

(1) Context-aware dynamics model (CaDM) [Lee et al. 2020]. This state-of-the-art method that decomposes the task of learning a global dynamics model into two stages: learning a context latent vector that captures the local dynamics and then predicting the next state conditioned on it.
(2) Autoencoder — ALPS w/o estimator and physics simulator.
(3) ALPS w/o the Fourier feature mapping — Fourier feature mapping is replaces with raw state trajectories $\tilde{x}_s$.
(4) ALPS w/o self-attention networks — self-attention network in the encoder is replaced by a simple MLP.

The models are compared based on three errors: state prediction error (SE), observation prediction error (OE) and parameter prediction error (PE) in solving the task of estimating physical parameters of three visual systems: pendulum, mass-spring-damper (MSD) and two-body. To generate a dataset, first, for each task initial state and physical parameters were sampled randomly, then 125 step rollout following the true system dynamics were rendered in a form of 64 by 64 by 3 pixel observation snapshots.

The performance evaluation results shows (Fig. 3) that ALPS reaches the best performance in predicting physical parameters utilizing both, self-attention and Fourier mapping. Moreover, Fourier feature mapping improves the results for pendulum, MSD, but two-body task, which proves using the mapping for systems that involve periodical or vibrational behavior. Regarding state predictions, ALPS achieves competitive results among the networks. In addition, it is visible that without self-attention networks, the SE increase drastically due to a large error in computing the velocity. Moreover, the high SE in the encoder network suggests that its latent representation is uninterpretable, which proves the idea of using physics to constrain the representation for estimating states in the encoder network. Lately, ALPS actives a comparable results in reconstructing observations.

5 ALPS LIMITATIONS

The first limitation that make it challenging to apply ALPS to more complicated systems such as contact and fluid dynamics is the requirement for system to have a dynamic equation coupled with an assumption that the system is linear. The solution for this challenge might be found in combining known physics with neural models. One of the possible approaches is using interaction networks [Battaglia et al. 2016] as a neural model. The model can reason about how objects in complex systems interact, including dynamical predictions. For physical reasoning, the model takes a graph of objects and relations as input [Scarselli et al. 2009], reasons about their interactions, and applies the effects and physical dynamics to predict new states. Another approach is to use a framework for unsupervised meta-learning of hybrid latent dynamics [Ye et al. 2024] (Meta-HyLaD). The Meta-HyLaD consists of two parts: the first part is integrating established mathematical expressions representing prior physics with neural functions that capture unknown errors to form a latent dynamic function; the second part is employing a meta-learning framework aimed at discerning and separately characterizing both components within the hybrid dynamics.

The second limitation is the requirement of neural ODE solver when simulating trajectories, which in addition results in high computational costs. The solution to the challenge might be found in using other technique to insert prior knowledge [Botev et al. 2021]. Using Hamiltonian Generative Networks (HGN) [Toth et al. 2020] and Lagrangian Generative Networks (LGN) [Lutter et al. 2019] as priors involves parameterizing kinetic and potential energies with neural networks. These networks, along with the encoder's latent state, enable simulating motion equations using numerical integrators like leap-frog for HGN and adaptive stepsize Runge-Kutta for LGN. Another approach is using Recurrent Generative Network (RGN) [Chen et al. 2019] that uses the same MLP as the Neural ODE model but parametrizes the discrete state evolution in latent space.

6 FUTURE WORK

One possible direction for future work is to study the self-attention mechanism more precisely. By visualizing attention weights in a simple vehicle steering dynamics scenario with the lateral path deviation x^p and turning rate x^r, it might be observed that the network primarily utilizes observations at the beginning and end of attention, suggesting a prediction of average velocity. Moreover, by analyzing the experiment of estimating x^r from x^p and vice versa. In the first case, the network attends to the neighboring observations at the current time (diagonal pattern) and performs derivative operation (local context). In the second case (Fig. 4) the network attends to the observations from beginning to the current time (triangular pattern) and performs integration operation (global context). Another possible direction is an extension of ALPS to differentiable rendering engines.

7 CONCLUSIONS

In this article we addressed the problem of predicting states and physical parameters of a system from observations with dynamic equations.

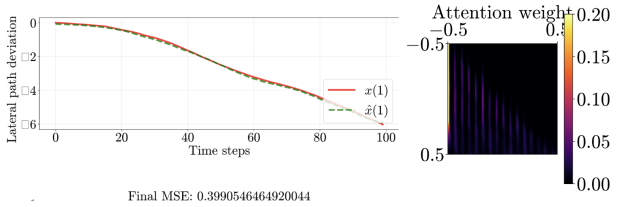

Fig. 4. Estimation of literal path deviation from the turning rate

REFERENCES

Peter W. Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, and Koray Kavukcuoglu. 2016. Interaction networks for learning about objects, relations and physics. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016*, Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett (Eds.). Barcelona, Spain, 4502–4510.

Aleksandar Botev, Andrew Jaegle, Peter Wirnsberger, Daniel Hennes, and Irina Higgins. 2021. Which priors matter? Benchmarking models for learning latent dynamics. *arXiv preprint arXiv:2111.05458* (Nov 2021). https://doi.org/10.48550/arXiv.2111.05458 arXiv:2111.05458 [stat.ML] Submitted on 9 Nov 2021.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. 2018. Neural Ordinary Differential Equations. In *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Eds.), Vol. 31. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2018/file/69386f6bb1dfed68692a24c8686939b9-Paper.pdf

Zhengdao Chen, Jianyu Zhang, Martin Arjovsky, and Léon Bottou. 2019. Symplectic Recurrent Neural Networks. *arXiv preprint arXiv:1909.13334* (2019). https://doi.org/10.48550/arXiv.1909.13334 Journal reference: 8th International Conference on Learning Representations (ICLR 2020).

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 770–778. https://doi.org/10.1109/CVPR.2016.90

Arthur Jacot, Franck Gabriel, and Clement Hongler. 2018. Neural Tangent Kernel: Convergence and Generalization in Neural Networks. In *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Eds.), Vol. 31. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2018/file/5a4be1fa34e62bb8a6ec6b91d2462f5a-Paper.pdf

Kimin Lee, Younggyo Seo, Seunghyun Lee, Honglak Lee, and Jinwoo Shin. 2020. Context-aware Dynamics Model for Generalization in Model-Based Reinforcement Learning. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*. JMLR: W&CP, 5757–5766. https://doi.org/10.48550/arXiv.2005.06800

Michael Lutter, Christian Ritter, and Jan Peters. 2019. Deep Lagrangian Networks: Using Physics as Model Prior for Deep Learning. In *International Conference on Learning Representations (ICLR)*. ICLR, Technische Universität Darmstadt, Darmstadt, Germany. https://doi.org/10.48550/arXiv.1907.04490 Published as a conference paper at ICLR 2019.

Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. 2019. On the Spectral Bias of Neural Networks. 5301–5310. http://proceedings.mlr.press/v97/rahaman19a.html

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2009. The graph neural network model. *IEEE Transactions on Neural Networks* 20, 1 (2009), 61–80.

Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. 2020. Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains. In *Advances in Neural Information Processing Systems (NeurIPS)*. Curran Associates, Inc. http://arxiv.org/abs/2006.10739v1

Peter Toth, Danilo Jimenez Rezende, Andrew Jaegle, Sébastien Racanière, Aleksandar Botev, and Irina Higgins. 2020. Hamiltonian Generative Networks. In *Proceedings of International Conference on Learning Representations (ICLR)*. https://doi.org/10.48550/arXiv.1909.13789 Submitted on 30 Sep 2019 (v1), last revised 14 Feb 2020 (this version, v2).

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

Tsung-Yen Yang, Justinian Rosca, Karthik Narasimhan, and Peter J Ramadge. 2022. Learning Physics Constrained Dynamics Using Autoencoders. In *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (Eds.), Vol. 35. Curran Associates, Inc., 17157–17172. https://proceedings.neurips.cc/paper_files/paper/2022/file/6d5e035724687454549b97d6c805dc84-Paper-Conference.pdf

Yubo Ye, Sumeet Vadhavkar, Xiajun Jiang, Ryan Missel, Huafeng Liu, and Linwei Wang. 2024. Unsupervised Learning of Hybrid Latent Dynamics: A Learn-to-Identify Framework. https://doi.org/10.48550/arXiv.2403.08194 arXiv:2403.08194 [cs.LG] Under Review.