A simple equivariant machine learning method for dynamics based on scalars

Weichi Yao,[*] **Kate Storey-Fisher**,[§] **David W. Hogg**,[†][‡] **& Soledad Villar**[§][¶]

Abstract

Physical systems obey strict symmetry principles. We expect that machine learning methods that intrinsically respect these symmetries should have higher prediction accuracy and better generalization in prediction of physical dynamics. In this work we implement a principled model based on invariant scalars [23], and release open-source code. We apply this *Scalars* method to a simple chaotic dynamical system, the springy double pendulum. We show that the Scalars method outperforms state-of-the-art approaches for learning the properties of physical systems with symmetries, both in terms of accuracy and speed. Because the method incorporates the fundamental symmetries, we expect it to generalize to different settings, such as changes in the force laws in the system.

1 Introduction

The laws of physics are coordinate-free and equivariant with respect to rotations, translations, and other kinds of transformations. Loosely speaking, a model is equivariant with respect to a group G (such as the rotation group, for example) if the transformation of all the inputs by a group element g leads to an output that is transformed by that same group element. A model is coordinate-free if a change of coordinates applied to all inputs leads to an output that has that same change of coordinates applied. Enforcing fundamental symmetries is crucial in the physical sciences. For instance, encoding rotational symmetries significantly improves predictions in molecular dynamics [1, 25, 20]; quantum energy levels on lattices obey symmetries that should be enforced in analysis tasks [5, 22, 17]; many problems in cosmology also rely on spatial symmetries, such as the translational and rotational invariance of weak lensing maps [29, 13]; and it has been noticed that enforcing relevant symmetries can improve turbulent flow prediction accuracy compared to the generic machine-learning methods [16, 4, 18] that do not naturally incorporate symmetries but only gradually improve the approximation.

Figure 1: The springy double pendulum used in this work.

There are different ways to enforce symmetries in machine learning models in general [6, 27]. In the simplest case—if we have enough training data—a sufficiently flexible model will naturally learn symmetric laws when the prediction task is in fact symmetric. If there are no enough data to learn the symmetries, one could consider adding training data rotated and translated (say) copies of training data elements; this is known as data augmentation [14, 2, 8, 11]. Another path for the enforcement of symmetries is to consider the composition of linear equivariant layers on tensor inputs with non-linear compatible activation functions; this typically requires irreducible representations [15, 21, 10] and code implementations are available for some examples but not for general symmetries. Recent work by [24] builds on the methods of [28, 26, 7] and enforces symmetries (such as rotation and scale) on convolutional neural networks to improve the prediction of physical dynamics. Another approach is to parametrize the space of equivariant linear functions by writing the corresponding constraints that the equivariance enforces, and then solve a large linear system [9].
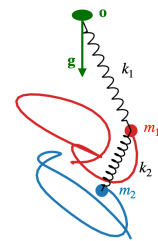
Recently, a new approach to enforcing physical symmetries in machine learning models was introduced in [23] that is simple and powerful compared to existing approaches. It proposes to construct explicitly invariant features (scalars) and use these as inputs to the model, in a way that is universally approximating. The method connects to the way classical physicists represent physical law in terms of coordinate-free scalar, vector, and tensor forms. However, the scalars approach has no publicly available implementation yet. In this work, we implement this method and apply it to learning the dynamics of a springy double pendulum, which is a standard problem in this area. We show that the scalars approach achieves state-of-the-art accuracy for this problem while being considerably faster and simpler than other approaches.

2 Modeling invariant and equivariant functions with scalars

In this work we focus on learning functions that are invariant or equivariant with respect to the orthogonal group $\mathrm{O}(d) = \{\mathbf{R} \in \mathbb{R}^{d \times d} : \mathbf{R}^\top \mathbf{R} = I\}$. The fundamental theorem of invariant functions for $\mathrm{O}(d)$ states that a function $\mathtt{f} : (\mathbb{R}^d)^n \to \mathbb{R}$ is $\mathrm{O}(d)$-invariant (i.e. $\mathtt{f}(\mathbf{R}\mathbf{v}_1, \dots, \mathbf{R}\mathbf{v}_n) = \mathtt{f}(\mathbf{v}_1, \dots, \mathbf{v}_n)$ for all $\mathbf{R} \in \mathrm{O}(d)$ and all $\mathbf{v}_i \in \mathbb{R}^d$) if and only if $\mathtt{f}$ can be expressed as a function of the scalar products of the input vectors:

$$\mathtt{f}(\mathbf{v}_1, \dots, \mathbf{v}_n) = \tilde{\mathtt{f}}\left((\mathbf{v}_i^\top \mathbf{v}_j)_{i,j=1}^n\right). \tag{1}$$

The formulation in [23] extends this characterization to equivariant vector functions, stating that a function $\mathtt{h} : (\mathbb{R}^d)^n \to \mathbb{R}^d$ is $\mathrm{O}(d)$-equivariant (i.e. $\mathtt{h}(\mathbf{R}\mathbf{v}_1, \dots, \mathbf{R}\mathbf{v}_n) = \mathbf{R}\mathtt{h}(\mathbf{v}_1, \dots, \mathbf{v}_n)$ for all $\mathbf{R}$ and $\mathbf{v}_i$) if and only if $\mathtt{h}$ can be expressed as a linear combination of the input vectors, where the coefficients are scalar invariant functions of the inputs:

$$\mathtt{h}(\mathbf{v}_1, \dots, \mathbf{v}_n) = \sum_{s=1}^n \tilde{\mathtt{f}}_s\left((\mathbf{v}_i^\top \mathbf{v}_j)_{i,j=1}^n\right) \mathbf{v}_s. \tag{2}$$

We construct $\mathrm{O}(d)$-invariant (equivariant) neural networks using multi-layer perceptrons (MLPs) that take the scalar products of the input vectors as input features. These neural networks are invariant (equivariant) by construction, and can universally approximate any invariant (equivariant) function since MLPs are universal.

3 Application: Springy double pendulum

We consider the dissipationless spherical double pendulum with springs, with a pivot o and two masses connected by springs (Figure 1). The kinetic energy $\mathcal{T}$ and potential energy $\mathcal{U}$ of the system are given by

$$\mathcal{T} = \frac{|\mathbf{p}_1|^2}{2m_1} + \frac{|\mathbf{p}_2|^2}{2m_2}, \tag{3}$$

$$\mathcal{U} = \frac{1}{2}k_1(|\mathbf{q}_1 - \mathbf{q}_o| - l_1)^2 + \frac{1}{2}k_2(|\mathbf{q}_2 - \mathbf{q}_1| - l_2)^2 - m_1\,\mathbf{g} \cdot (\mathbf{q}_1 - \mathbf{q}_o) - m_2\,\mathbf{g} \cdot (\mathbf{q}_2 - \mathbf{q}_o), \tag{4}$$

where $\mathbf{q}_1, \mathbf{p}_1$ are the position and momentum vectors for mass m_1, similarly $\mathbf{q}_2, \mathbf{p}_2$ for mass m_2, and a position $\mathbf{q}_o$ for the pivot. The springs have scalar spring constants k_1, k_2, and natural lengths l_1, l_2. The gravitational acceleration vector is $\mathbf{g}$.

We consider the task of learning the dynamics of the pendulum; the goal is to predict its trajectory at later times from different initial states. For this task, the parameters of the model $m_1, m_2, k_1, k_2, l_1, l_2$ are fixed during training and test but unknown. Our training inputs are N different initializations at $t_0 = 0$ of the pendulum positions and momenta, and the labels are the positions and momenta at a set of T later times t:

$$\mathbf{z}^i(t) = (\mathbf{q}_1^i(t), \mathbf{q}_2^i(t), \mathbf{p}_1^i(t), \mathbf{p}_2^i(t)), \quad t \in \{t_0 = 0, t_1, \dots, t_T\} \text{ and } i \in \{1, \dots, N\}. \tag{5}$$

To this end we aim to learn the function f that predicts the dynamics:

$$\begin{aligned} f : (\mathbb{R}^3)^4 \times \mathbb{R}^3 \times \mathbb{R}^3 \times \mathbb{R} &\to (\mathbb{R}^3)^4 \\ (\mathbf{z}(0), \mathbf{q}_o, \mathbf{g}, \Delta t) &\mapsto \hat{\mathbf{z}}(\Delta t). \end{aligned} \tag{6}$$

This function is O(3)-equivariant in both position and momentum, and translation equivariant in position, namely, for all $\mathbf{R} \in O(3)$ and all $\mathbf{w} \in \mathbb{R}^3$ we have

$$f((\mathbf{Rq}_1 + \mathbf{w}, \mathbf{Rq}_2 + \mathbf{w}, \mathbf{Rp}_1, \mathbf{Rp}_2), \mathbf{Rq}_o + \mathbf{w}, \mathbf{Rg}) = \mathbf{R}f((\mathbf{q}_1, \mathbf{q}_2, \mathbf{p}_1, \mathbf{p}_2), \mathbf{q}_o, \mathbf{g}) + \mathbf{w}. \quad (7)$$

This system can be seen as O(2)-equivariant [9] (with the gravitational acceleration breaking the full O(3) symmetry), but we prefer to see it as O(3)-equivariant, with the gravitational acceleration treated as an input to the model. In particular, if we treat it this way, we might be able to train with one value of $\mathbf{g}$ and generalize at test time to other values of $\mathbf{g}$.

4 Learning dynamics from data

The double pendulum with springs is a Hamiltonian dynamical system, where $\mathcal{H} = \mathcal{T} + \mathcal{U}$ is time invariant. Inspired by [9] we consider two approaches to model f in equation (6): Neural ordinary differential equations (N-ODEs) [3], and Hamiltonian neural networks (HNNs) [12, 19]. Both approaches, depicted in Figure 2, are supervised with the training data described in equation (5) and loss function $\ell(\theta) = \sum_{i=1}^{N} \sum_{j=0}^{T} \|\mathbf{z}^i(t_j) - \hat{\mathbf{z}}^i(t_j)\|^2$, where θ represents the parameters in the neural networks to optimize. We implement HNNs and N-ODEs with scalar-based MLPs following the idea from [23] described in Section 2.

N-ODEs use neural networks to parameterize the derivative function $F(\mathbf{z}(t), t) = \frac{\mathrm{d}\mathbf{z}(t)}{\mathrm{d}t}$ [3]. Here,

$$\frac{\mathrm{d}\mathbf{v}}{\mathrm{d}t} = F\big((\mathbf{q}_1, \mathbf{q}_2, \mathbf{p}_1, \mathbf{p}_2), \mathbf{q}_o, \mathbf{g}\big), \quad (8)$$

since the dynamics are time-homogeneous. The vector function F is O(3)-equivariant with respect to all its inputs, and translation invariant in position. Therefore, there exists a unique function F_o such that $F(\mathbf{q}_1, \mathbf{p}_1, \mathbf{q}_2, \mathbf{p}_2, \mathbf{q}_o, \mathbf{g}) = F_o(\mathbf{q}_1 - \mathbf{q}_o, \mathbf{q}_2 - \mathbf{q}_o, \mathbf{q}_1 - \mathbf{q}_2, \mathbf{p}_1, \mathbf{p}_2, \mathbf{g})$. Note that the input $\mathbf{q}_1 - \mathbf{q}_2$ is redundant, we add it for conceptual reasons, see equation (4). We model F_o as $\mathbf{h}$ in (2) with vector inputs $\mathcal{E} := \{\mathbf{q}_1 - \mathbf{q}_o, \mathbf{q}_2 - \mathbf{q}_o, \mathbf{q}_1 - \mathbf{q}_2, \mathbf{p}_1, \mathbf{p}_2, \mathbf{g}\}$, obtaining

$$F_o\big((\mathbf{q}_1, \mathbf{p}_1, \mathbf{q}_2, \mathbf{p}_2), \mathbf{q}_o, \mathbf{g}\big) = \sum_{\mathbf{u} \in \mathcal{E}} g_\mathbf{u}\big(\{\sigma(\mathbf{e}^\top \mathbf{e}') : \forall \mathbf{e}, \mathbf{e}' \in \mathcal{E}, \ \sigma \in \Omega\}\big)\, \mathbf{u}, \quad (9)$$

where Ω is a set of scalar transformation functions (e.g. $\Omega = \{\sigma_1 : x \mapsto x; \ \sigma_2 : x \mapsto \sqrt{|x|}\}$). Here the set of functions $\{g_\mathbf{u} : \mathbf{u} \in \mathcal{E}\}$ are scalar-based and approximated by MLPs. The predicted "rollout" trajectory, $\big(\mathbf{z}(0), \hat{\mathbf{z}}(t_1), \ldots, \hat{\mathbf{z}}(t_T)\big)$, is computed iteratively with an ODE solver,

$$\hat{\mathbf{z}}(t_j) = \text{ODESolve}\big(\hat{\mathbf{z}}(t_{j-1}), t_{j-1}, t_j, F_o\big), \quad \hat{\mathbf{z}}(0) = \mathbf{z}(0), \quad (10)$$

for each time step $j = 1, \ldots, T$.

For the HNN case, we parameterize the Hamiltonian function of the dynamical system with a neural network, and we use a Hamiltonian integrator to predict the dynamics [12]. The Hamiltonian $\mathcal{H}$ is an invariant scalar function, so similarly to the previous case, we can write $\mathcal{H}(\mathbf{q}_1, \mathbf{q}_2, \mathbf{p}_1, \mathbf{p}_2, \mathbf{q}_o, \mathbf{g}) = \mathcal{H}_o(\mathbf{q}_1 - \mathbf{q}_o, \mathbf{q}_2 - \mathbf{q}_o, \mathbf{q}_1 - \mathbf{q}_2, \mathbf{p}_1, \mathbf{p}_2, \mathbf{g})$. We model $\mathcal{H}$ as $\mathbf{f}$ in (1), obtaining

$$\mathcal{H}(\mathbf{q}_1, \mathbf{p}_1, \mathbf{q}_2, \mathbf{p}_2, \mathbf{q}_o, \mathbf{g}) = h\big(\{\sigma(\mathbf{e}^\top \mathbf{e}') : \forall \mathbf{e}, \mathbf{e}' \in \mathcal{E}, \ \sigma \in \Omega\}\big), \quad (11)$$

where Ω is a set of scalar transformation functions as in the N-ODE case, h is a scalar function based on all the inner products of the vectors in $\mathcal{E}$. We approximate h by MLPs. The Hamiltonian dynamics are obtained by integrating the corresponding ODE, $\frac{\mathrm{d}\mathbf{z}(t)}{\mathrm{d}t} = \mathbf{J}\nabla\mathcal{H}$, where $\mathbf{J} = [\mathbf{0} \ \mathbf{I}; -\mathbf{I} \ \mathbf{0}]$, with an ODE solver as in equation (10) [19].

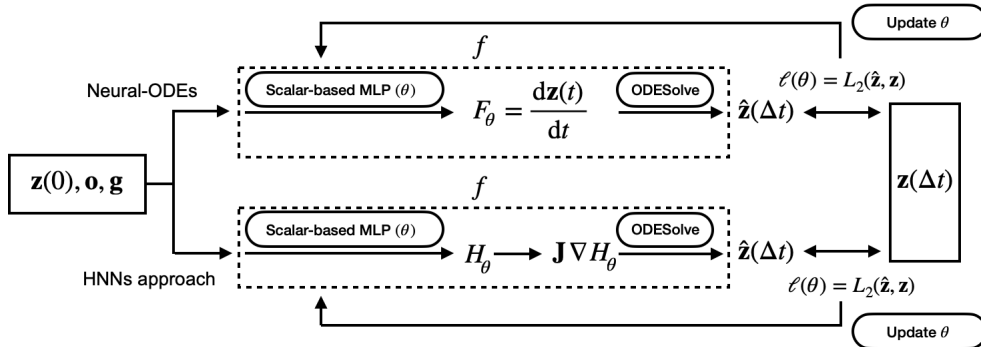Figure 2: The solving process of learning springy dynamics from data.

5 Numerical results

Our numerical experiments use the same data generation scheme as in [9]. The training data is generated from equation (5) with $T = 5$ and $N = 500$. We compare the scalar-based implementation with the constraint-based implementation of the symmetries considered in [9], i.e., O(2), the rotation group of order 2 (SO(2)), and the dihedral group of degree 2 and 6 (D_2 and D_6). We also compare to standard non-symmetry enforcing N-ODEs and HNNs models as a baseline. The source code is published in `https://github.com/weichiyao/ScalarEMLP`.
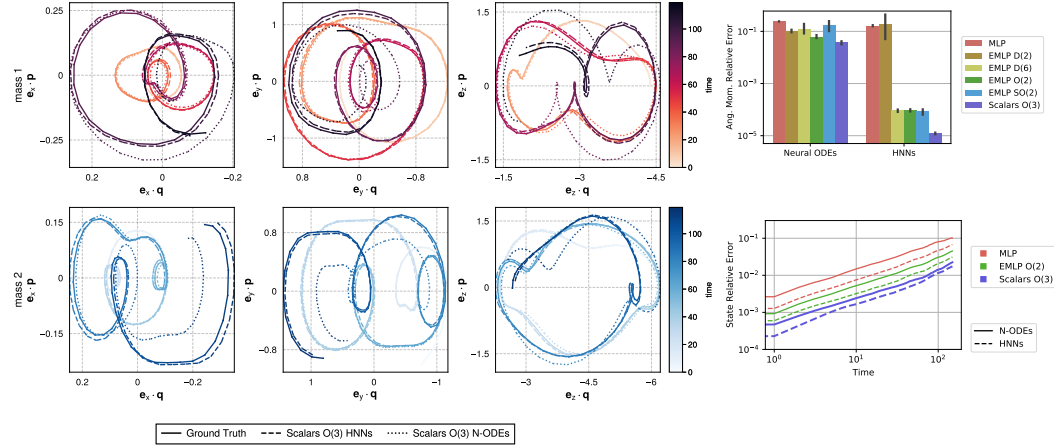
Figure 3: **Left**: Ground truth and predictions of mass 1 (top) and 2 (bottom) in the phase space w.r.t. each dimension. HNNs exhibits more accurate predictions for longer time scales. **Top right**: Invariance of the projection of the angular momentum onto the gravitational force vector $L_\perp$ as in (12). The geometric mean of the relative error is computed over $T = 150$ rollouts and averaged across initial conditions. Errorbars are 95% confidence interval over 3 runs. **Bottom right**: The state relative error along the orbit as in (13). Shaded regions show one standard deviation in log space across the different trajectories, showing the variance in the data. Only the best EMLP method is shown for clarity.

In Figure 3 we show the relative error of the different models in terms of the angular momentum defined as

$$L_\perp(t) = \left(\mathbf{q}_1(t) \times \mathbf{p}_1(t) + \mathbf{q}_2(t) \times \mathbf{p}_2(t)\right)^\top \cdot \frac{\mathbf{g}}{\|\mathbf{g}\|}, \tag{12}$$

the projection of the angular momentum onto $\mathbf{g}$. We observe, similar to [9], that the Hamiltonian-based models conserve the angular momentum as expected, whereas the N-ODEs do not. We also depict the relative state errors, defined at a given time t in terms of the positions and momenta of the masses,

$$\text{State.RelErr}(t) = \frac{\sqrt{(\hat{\mathbf{z}}(t) - \mathbf{z}(t))^\top (\hat{\mathbf{z}}(t) - \mathbf{z}(t))}}{\sqrt{\hat{\mathbf{z}}(t)^\top \hat{\mathbf{z}}(t)} + \sqrt{\mathbf{z}(t)^\top \mathbf{z}(t)}}, \tag{13}$$

of the scalar-based models along the orbits in Figure 3. Table 1 gives the geometric mean of these state relative errors over $T = 150$. It shows that the scalars approach significantly outperforms the baseline MLP models and the EMLP methods, for both the N-ODE and HNN cases.

6 Discussion

This paper provides a proof of concept implementation for scalar-based equivariant machine learning. Our experimental results for the springy pendulum problem support the intuition that using correct knowledge about the fundamental symmetries of the system leads to better performance on the machine learning algorithms. This not only applies to O(3) and translation equivariance, but also the Hamiltonian symmetry, as it is apparent in Figure 3.

The simplicity of the scalar-based formulation results in fast methods with state of the art performance. However, many technical challenges remain open, for instance establishing the robustness

	Scalars O(3)	EMLP				MLP
		O(2)	SO(2)	D_2	D_6	
N-ODEs:	**.009 ± .001**	.020 ± .002	.051 ± .036	.023 ± .002	.036 ± .025	.048 ± .000
HNNs:	**.005 ± .002**	.012 ± .002	.016 ± .003	.111 ± .167	.013 ± .002	.028 ± .001

Table 1: Geometric mean (standard deviation computed over 3 trials) of state relative errors of the springy pendulum over $T = 150$. Results are shown for the scalar-based MLP HNNs and N-ODEs vs EMLP models from [9] with different symmetry groups, and the standard MLP with no symmetry enforcement. EMLP trains in about one hour, whereas the scalars-based method takes 15 minutes (same machine, same training data).

of the approach under noisy data, making the method scalable with respect to the number of input vectors, and investigating the ability of the model to generalize to other settings. One example of the latter is inferring the behavior of the springy pendulum when the masses or natural lengths change. To this end it may be interesting to combine the scalars-based modeling with symbolic regression.

References

[1] B. Anderson, T. S. Hy, and R. Kondor. Cormorant: Covariant molecular neural networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[2] A. K. Aniyan and K. Thorat. Classifying radio galaxies with the convolutional neural network. *The Astrophysical Journal Supplement Series*, 230(2):20, Jun 2017.

[3] R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[4] B. Chidester, M. N. Do, and J. Ma. Rotation equivariance and invariance in convolutional neural networks. *arXiv preprint arXiv:1805.12301*, 2018.

[5] K. Choo, G. Carleo, N. Regnault, and T. Neupert. Symmetries and many-body excitations with neural-network quantum states. *Physical Review Letters*, 121:167204, Oct 2018.

[6] T. Cohen, M. Weiler, B. Kicanaoglu, and M. Welling. Gauge equivariant convolutional networks and the icosahedral cnn. In *International Conference on Machine Learning*, pages 1321–1330. PMLR, 2019.

[7] T. S. Cohen and M. Welling. Group equivariant convolutional networks. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, page 2990–2999, 2016.

[8] H. Domínguez Sánchez, M. Huertas-Company, M. Bernardi, D. Tuccillo, and J. L. Fischer. Improving galaxy morphologies for sdss with deep learning. *Monthly Notices of the Royal Astronomical Society*, 476(3):3661–3676, Feb 2018.

[9] M. Finzi, M. Welling, and A. G. Wilson. A practical method for constructing equivariant multilayer perceptrons for arbitrary matrix groups. *arXiv:2104.09459*, 2021.

[10] F. Fuchs, D. Worrall, V. Fischer, and M. Welling. Se (3)-transformers: 3d roto-translation equivariant attention networks. *Neural Information Processing Systems*, 33, 2020.

[11] R. González, R. Muñoz, and C. Hernández. Galaxy detection and identification using deep learning and data augmentation. *Astronomy and Computing*, 25:103–109, 2018.

[12] S. Greydanus, M. Dzamba, and J. Yosinski. Hamiltonian neural networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, 2019.

[13] A. Gupta, J. M. Z. Matilla, D. Hsu, and Z. Haiman. Non-gaussian information from weak lensing data via deep learning. *Physical Review D*, 97(10), May 2018.

[14] M. Huertas-Company, R. Gravet, G. Cabrera-Vives, P. G. Pérez-González, J. S. Kartaltepe, G. Barro, M. Bernardi, S. Mei, F. Shankar, P. Dimauro, and et al. A catalog of visual-like morphologies in the 5 candels fields using deep learning. *The Astrophysical Journal Supplement Series*, 221(1):8, Oct 2015.

[15] R. Kondor. N-body networks: a covariant hierarchical neural network architecture for learning atomic potentials. *arXiv:1803.01588*, 2018.

[16] J. Ling, A. Kurzawskim, and J. Templeton. Reynolds averaged turbulence modeling using deep neural networks with embedded invariance. *Journal of Fluid Mechanics*, pages 155–166, 2017.

[17] D. Luo, Z. Chen, K. Hu, Z. Zhao, V. M. Hur, and B. K. Clark. Gauge Invariant Autoregressive Neural Networks for Quantum Lattice Models. *arXiv:2101.07243*, 2021.

[18] M. Mattheakis, P. Protopapas, D. Sondak, M. D. Giovanni, , and E. Kaxiras. Physical symmetries embedded in neural networks. *arXiv preprint arXiv:1904.08991*, 2019.

[19] A. Sanchez-Gonzalez, V. Bapst, K. Cranmer, and P. Battaglia. Hamiltonian graph networks with ode integrators, 2019.

[20] K. T. Schütt, O. T. Unke, and M. Gastegger. Equivariant message passing for the prediction of tensorial properties and molecular spectra, 2021.

[21] N. Thomas, T. Smidt, S. Kearnes, L. Yang, L. Li, K. Kohlhoff, and P. Riley. Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds. *arXiv:1802.08219*, 2018.

[22] T. Vieijra, C. Casert, J. Nys, W. De Neve, J. Haegeman, J. Ryckebusch, and F. Verstraete. Restricted boltzmann machines for quantum states with non-abelian or anyonic symmetries. *Physical Review Letters*, 124(9), Mar 2020.

[23] S. Villar, D. W. Hogg, K. Storey-Fisher, W. Yao, and B. Blum-Smith. Scalars are universal: Equivariant machine learning, structured like classical physics. *CoRR*, abs/2106.06610, 2021.

[24] R. Wang, R. Walters, and R. Yu. Incorporating symmetry into deep dynamics models for improved generalization. In *International Conference on Learning Representations ICLR*, 2021.

[25] Z. Wang, C. Wang, S. Zhao, S. Du, Y. Xu, B.-L. Gu, and W. Duan. Symmetry-adapted graph neural networks for constructing molecular dynamics force fields, 2021.

[26] M. Weiler and G. Cesa. General e(2)-equivariant steerable cnns. In *Advances in Neural Information Processing Systems*, volume 32, pages 14334–14345, 2019.

[27] M. Weiler, P. Forré, E. Verlinde, and M. Welling. Coordinate independent convolutional networks–isometry and gauge equivariant convolutions on riemannian manifolds. *arXiv preprint arXiv:2106.06020*, 2021.

[28] D. Worrall and M. Welling. Deep scale-spaces: Equivariance over scale. In *Advances in Neural Information Processing Systems*, volume 32, pages 7364–7376, 2019.

[29] J. Yoo, N. Grimm, E. Mitsou, A. Amara, and A. Refregier. Gauge-invariant formalism of cosmological weak lensing. *Journal of Cosmology and Astroparticle Physics*, 2018(04):029–029, Apr 2018.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. or example:

- Did you include the license to the code and datasets? [Yes] See Section .
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] This does not apply to our work.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

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]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (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)? [Yes] Training of scalar-based multilayer perceptrons is easy and fast. We provide further details in `https://github.com/weichiyao/ScalarEMLP`.

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

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]