Extreme Mechanics Letters 58 (2023) 101925

Contents lists available at ScienceDirect

Extreme Mechanics Letters

journal homepage: www.elsevier.com/locate/eml

Rapidly encoding generalizable dynamics in a Euclidean symmetric neural network

Qiaofeng Li [a,b,c], Tianyi Wang [b], Vwani Roychowdhury [b,*], M. Khalid Jawed [a,*]

[a] Department of Mechanical and Aerospace Engineering, University of California, Los Angeles, 90095, CA, USA
[b] Department of Electrical and Computer Engineering, University of California, Los Angeles, 90095, CA, USA
[c] Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, 02139, MA, USA

ARTICLE INFO

Article history:
Received 5 August 2022
Received in revised form 24 October 2022
Accepted 9 November 2022
Available online 14 November 2022

Keywords:
Reduced-order model
Data-driven
Deep learning
Neural network
Neural ordinary differential equation

ABSTRACT

Slinky, a helical elastic rod, is a seemingly simple structure with unusual mechanical behavior; for example, it can walk down a flight of stairs under its own weight. Taking Slinky as a test-case, we propose a physics-informed deep learning approach for building reduced-order models of physical systems. The approach introduces a Euclidean symmetric neural network (ESNN) architecture that is trained under the neural ordinary differential equation framework to learn the 2D latent dynamics from the motion trajectory of a reduced-order representation of the 3D Slinky. The ESNN implements a physics-guided architecture that simultaneously preserves energy invariance and force equivariance under Euclidean transformations of the input, including translation, rotation, and reflection. The embedded Euclidean symmetry provides physics-guided interpretability and generalizability, while preserving the full expressive power of the neural network. We demonstrate that the ESNN approach is able to accelerate simulation by one to two orders of magnitude compared to traditional numerical methods and achieve a superior generalization performance while classic neural networks fail to learn the Slinky dynamics, i.e., the ESNN, trained on a single demonstration case, predicts the motions accurately for unseen cases of different Slinky configurations and boundary conditions. Further investigation into the ESNN reveals that it explicitly learns the nonlinear coupling between stretching and bending of the Slinky.

1. Introduction

Reduced-order models (ROMs) [1–4] are simplifications of high-order complex models that are often derived from first principles. ROMs, through ingenious reduction of degrees of freedom (DoFs), gain significant advantages in computation cost compared with high-order complex models without significant loss of accuracy. Their application is crucial in scenarios where the computation is expensive [5], repeated computations are required [6], or the application is computation-time sensitive [7,8]. Traditionally ROMs are built by experts with domain-knowledge through repeated trial-and-error and time-consuming analysis. In this paper, we seek a data-driven approach to automatically construct ROMs from observed data. Recently, deep learning based on deep neural networks [9,10] (DNNs) has demonstrated two critical features making it a perfect candidate technique for automatic ROM construction: on one hand, the data-driven models offer possibilities for superior computation efficiency compared to classic methods [11–19]; on the other hand, it enables advances towards human-like, automated rule discovery and learning of the dynamics of a system from observations [20–26].

Unconstrained neural networks (NNs), trained to directly fit time-series data generated by dynamical systems, however, often cannot replicate the dynamics under unseen initial and boundary conditions because they fail to learn the underlying physical constraints, such as Euclidean symmetry of space and conservation of energy. Indeed, recent works have shown that incorporating such constraints into deep learning is vital for generalization ability, learning efficiency, and interpretability [27,28]. For example, Hamiltonian Neural Network [29] and Lagrangian Neural Network [30] introduced the physical prior of energy conservation to neural networks for learning dynamics. Geometric deep learning [28,31] and equivariant neural networks [32–36] leverage geometric symmetry properties of the network input to improve the quality of inherent knowledge learned by neural networks. Successful applications include drug discovery [37,38], quantum chemistry [39], and particle physics [40,41].

In this paper, we introduce a physics-informed deep learning approach to learning 2D reduced-order dynamics of a Slinky. Instead of a long elastic helix in 3D space, a Slinky is described

* Corresponding authors.
E-mail addresses: vwani@ee.ucla.edu (V. Roychowdhury), khalidjm@seas.ucla.edu (M.K. Jawed).

https://doi.org/10.1016/j.eml.2022.101925
2352-4316/

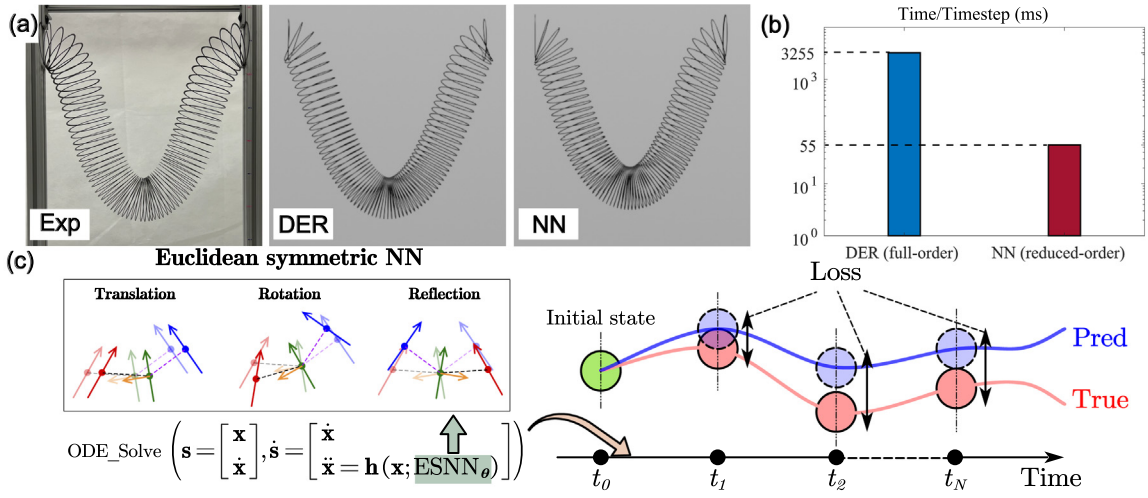

Fig. 1. (color online). Overview of the proposed Euclidean symmetric neural network (ESNN). (a) Static deformation comparison across a real-world Slinky experiment, the discrete elastic rod (DER) simulation, and the reduced-order model based on the ESNN. (b) Computation time comparison between DER simulation and the proposed NN method. (c) The core features of the proposed NN method: the NN embeds energy invariance and force equivariance on Euclidean transformations of the input Slinky configuration, including translation, rotation, and reflection. The NN is trained under the neural ordinary differential equation framework.

as a series of connected bars in a 2D plane [42]. The approach is guided by two principles: (i) The system trajectories along time are constructed by integrating the ordinary differential equation (ODE) formulated by Newton's second law of motion; (ii) A neural network is trained to predict the 2D surrogate forces on the bars so that the observed trajectories can be generated following the ODE. We endow the neural force predictor with Euclidean symmetry and energy conservation and propose an Euclidean symmetric neural network (ESNN) architecture. In addition, the neural force predictor is faced with the challenge of compounding error, i.e., the error in force prediction can accumulate through integration, leading to an erroneous simulated trajectory that seriously deviates from reality. We address this by going beyond learning separated state-force pairs and use the neural ordinary differential equation (NODE) framework [43] to match the entire trajectory. Under such framework, the consequence of a force prediction error in later time steps is considered in the loss, encouraging the neural network to be prescient and not to overfit proximate dynamics.

We show that this approach is able to accurately predict the motion of a real-world Slinky, and by learning from a single demonstration case, to generalize to unseen Slinky configurations, boundary conditions, and even a softer Slinky by exploiting "physics" – scaling the neural force predictor with the material stiffness. The computational speed is increased by roughly 60 times compared with state-of-the-art 3D simulation method (discrete elastic rod, DER [44], also see Supplementary Information S1 and Fig. S1) due to the significantly reduced DoFs (Fig. 1). To the best of the authors' knowledge, we for the first time demonstrate, with experimental validation, that a deep learning approach is capable of extensively generalizing from one single learning case, and that a complex system in the real world could benefit from this reality-virtual-reality closed-loop pipeline.

2. Results

2.1. The Slinky triplet formulation

The 2D Slinky representation is shown in Fig. 2(a). The top and bottom vertices of a 3D Slinky cycle are projected onto the XOY plane. The vector pointing from the projected bottom vertex to the projected top vertex is the 2D representation of that cycle, referred to as a *bar* hereafter. The coordinates of the ith bar are $\mathbf{x}_i = [x_i, y_i, \alpha_i] \in \mathbb{R}^3$: x_i and y_i are the coordinates of the center point, and α_i is the angle of the bar with respect to the y axis. For a Slinky of N_c cycles, its 2D representation has N_c bars, and the states of these N_c bars along time constitute the 2D system trajectory of the Slinky. Such a 2D reduced-order representation retains the essential geometry of the 3D structure of a Slinky, based on which a 3D helical shape is still reconstructable (Supplementary Information S5) despite the substantial reduction in the number of DoFs.

2.2. The Euclidean symmetric neural network

Fig. 2(c) shows the workflow of the ESNN. Three adjacent bars are grouped into a *triplet* as an input unit. For a Slinky of N_c 2D bars, each bar is used as the central bar of a triplet, resulting in N_c triplets (N_c instead of $N_c - 2$ because of special boundary treatment. See Supplementary Information S6). Consider a single input of the NN, i.e., the global coordinates of the ith triplet $\vec{\xi}_i = [\mathbf{x}_{i-1}^{\mathrm{T}}, \mathbf{x}_i^{\mathrm{T}}, \mathbf{x}_{i+1}^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^9$, where the superscript T represents transpose operation. For simplicity, we omit the triplet index i and denote $\vec{\xi}_i$ as $\vec{\xi}$. The first step in the ESNN is to make the representation invariant to rigid body translation and rotation, by transforming the global coordinates $\vec{\xi}$ into relative coordinates $\mathbf{z} \in \mathbb{R}^6$ (Supplementary Information S4). Then $\mathbf{z}$ and its 3 reflected copies are passed through a DenseNet-like structure [45] parameterized by $\boldsymbol{\theta}$, denoted as $f_{\boldsymbol{\theta}}(\cdot)$. The outputs are 4 scalars, and their summation is the energy surrogate E:

$$E(\mathbf{z}) = f_{\boldsymbol{\theta}}(\mathbf{z}) + f_{\boldsymbol{\theta}}(R_x(\mathbf{z})) + f_{\boldsymbol{\theta}}(R_y(\mathbf{z})) + f_{\boldsymbol{\theta}}(R_x(R_y(\mathbf{z}))) \tag{1}$$

where $R_x(\cdot)$ and $R_y(\cdot)$ stand for the reflection of the triplet coordinates about x and y axes, respectively. Note that the reflections are self-inverse and commutative, i.e.,

$$R_x(R_x(\cdot)) = I(\cdot), \;\; R_y(R_y(\cdot)) = I(\cdot), \;\; \text{and } R_x(R_y(\cdot)) = R_y(R_x(\cdot))$$

where $I(\cdot)$ is the identity transformation. Therefore, the energy surrogate E satisfies the following property:

$$E(\mathbf{z}) = E(R_x(\mathbf{z})) = E(R_y(\mathbf{z})) = E(R_x(R_y(\mathbf{z}))) \tag{2}$$

That is, E is invariant to reflections on $\mathbf{z}$ and thus on $\vec{\xi}$; E is also invariant to rigid body transformation on $\vec{\xi}$ due to such invariance of $\mathbf{z}$. The output surrogate force on the middle bar of the triplet, $\mathbf{F} \in \mathbb{R}^3$, is generated by taking the derivative of E with respect

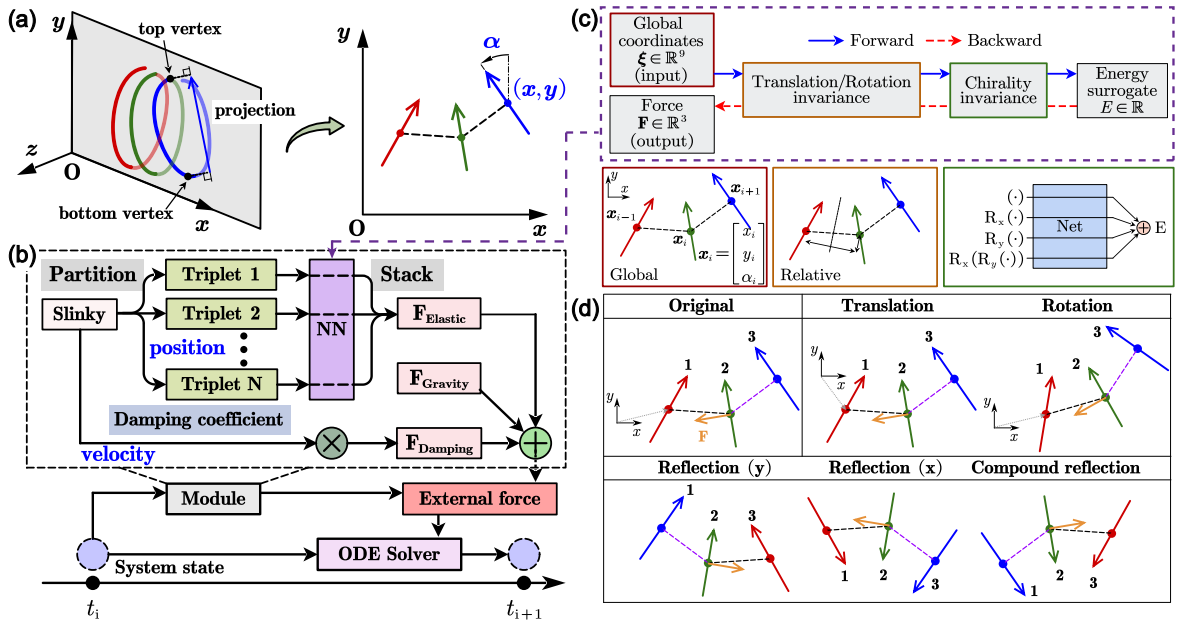

Fig. 2. (color online). Detailed schematic of the ESNN. (a) Construction of the 2D bars by projecting 3D Slinky cycles onto the XOY plane. (b) Time marching under the NODE framework. At time t_i, the Slinky system is partitioned into triplets and passed through the same ESNN. The output force vectors are stacked and fed into an ODE solver to update the system state at t_{i+1}. (c) Workflow of the ESNN. The "Net" in the chiral transformation module is a DenseNet-like neural network of 5 hidden layers (Supplementary Information S3). (d) Summary of the energy invariance and force equivariance properties of the ESNN. The original triplet configuration and its Euclidean transformed copies generate the same energy surrogate (invariance) and equivariant force vectors.

to $\mathbf{x}_i \in \mathbb{R}^3$ using the automatic differentiation mechanism in PyTorch [46], i.e., $\mathbf{F}(\vec{\xi}) = \partial E(\vec{\xi})/\partial \mathbf{x}_i$. It is straightforward to prove that $\mathbf{F}$ is equivariant to rigid body and chiral transformations on $\vec{\xi}$ (Fig. 2(d)).

The entire ESNN can be considered as a general function $\mathbf{F} = \mathrm{ESNN}_{\boldsymbol{\theta}}(\vec{\xi})$, with the aforementioned invariances and equivariances guaranteed regardless of the parameters $\boldsymbol{\theta}$. Conventionally, to obtain good performance with an unconstrained neural network for all orientation and chirality of the input, the approach is to augment the training dataset with various rotation and reflection. In comparison, the advantages of ESNN are trifold: (1) The Euclidean symmetry is strictly guaranteed (instead of only approximated in the dataset-augmentation approach); (2) Training cost is significantly lower due to the concise training dataset without symmetry augmentation; (3) A more compact NN architecture is possible since weights are dedicated to learning the functional form of the surrogate elastic energy instead of the symmetries. This bolsters the succeeding gain in computational speed.

2.3. The neural ordinary differential equation training

Fig. 2(b) illustrates how a 2D state is evolved from time t_i to t_{i+1}. At each time step, N_c triplets are constructed, each of which is passed through the ESNN. The output is N_c 3-dimensional vectors representing the predicted surrogate elastic forces on the middle bars of the triplets. These vectors are then concatenated to form the elastic force vector for the entire Slinky system. Then the system state can be updated by any ODE solver, 5th order Dormand–Prince method in this paper. By repeating this update procedure from a known initial state, a predicted trajectory for the reduced-order Slinky system is calculated. The ESNN weights $\boldsymbol{\theta}$ is optimized to minimize the mean-squared error loss comparing the whole predicated trajectory and the ground truth across multiple observation time steps (Fig. 1(c)). To enable efficient optimization, the gradient of such loss w.r.t. the weights is calculated by the adjoint method introduced in NODE [43]. The NODE training scheme naturally aligns the optimization objective with the ODE simulation framework and improves accuracy and stability of the 2D dynamics dictated by the ESNN. To minimize the loss, it requires not only that the ESNN predicts instantaneous surrogate forces with small errors, but also that the errors do not compound when the neural force function is integrated through an ODE solver.

We test the ESNN on a commercially-available 76-cycle Slinky (Poof-Slinky, Inc.) under large nonlinear deformation. The experiment setup is shown in Fig. 3(a). We record a single Slinky motion case using a camera and calibrate a 3D DER model [44,47–50] for comparison and 2D training data generation. Based on the calibrated model a 3D simulation is run with the same initial and boundary conditions as the experiment and with the damping removed. The 2D system trajectory is constructed based on the projection of 3D data. During ESNN training, the trajectory length is gradually increased from 2 to 70 to improve training efficiency. The simulation time step is 0.01 s, making the final training trajectory time span 0.7 s. Refer to Supplementary Information S7 and S8 for details on DER model calibration and ESNN training. A damping and a contact model [51] (Supplementary Information S2) are added in the ESNN deployment to match the real-world energy dissipation and non-penetration. The ESNN is trained on this one case, and tested on other unseen cases.

2.4. Generalization to unseen conditions

After 1000 training epochs, a simulation is run with the trained ESNN for 2.5 s. The predicted trajectory, of which the first 0.7 s span is called train phase and the 0.7–2.5 s span extrapolation phase, is compared with 3D DER simulation in Fig. 3(c). The ESNN results match the ground truth well not only within the train phase, but also in the extrapolation phase up to 2.5 s, i.e., more than 2 times the training time span. Note that extrapolation here refers to time-domain forecasting rather than making predictions on "out-of-distribution" data. The final static deformations given by the experiment, DER, and ESNN are in good agreement, as

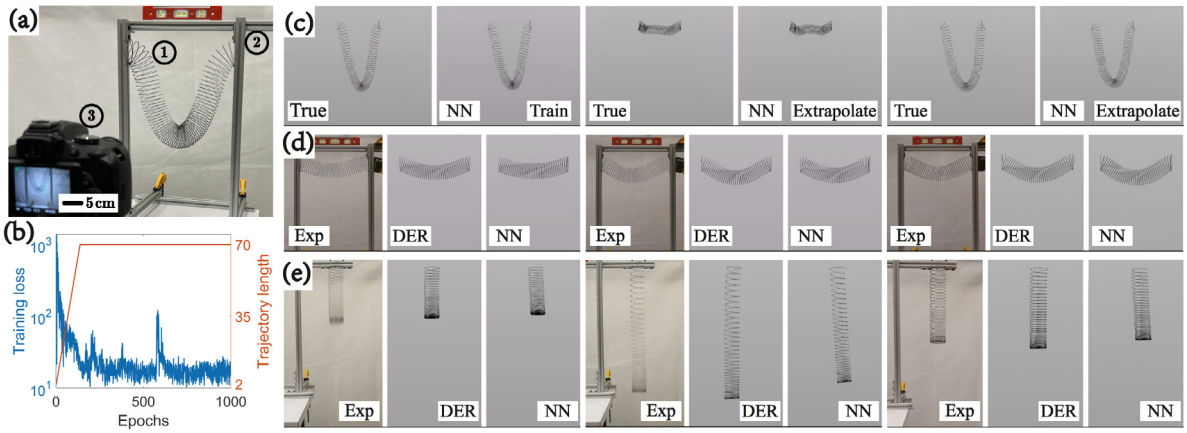

Fig. 3. (color online). Training and generalization results of the ESNN. (a) Experimental apparatus: a 76-cycle Slinky ① is supported by an experiment frame ②. A camera ③ records the Slinky motion. (b) The loss history of the NODE training for 1000 epochs. The training trajectory length is gradually increased from 2 to 70. (c) Comparison between 3D DER ground truth and reconstruction from the ESNN reduced-order model in train phase and extrapolation phase (Supplementary Movie S1). Time shots are at 0.45 s, 2.03 s, and 2.50 s (from left to right). (d) Comparison of an unseen 40-cycle Slinky across the experiment, 3D DER simulation, and reconstruction from the ESNN reduced-order model (Supplementary Movie S2). Time shots are at 0.53 s, 1.43 s, and static (from left to right). (e) Comparison of testing on a 40-cycle Slinky with a different boundary condition and orientation (Supplementary Movie S3). Time shots are at 0.2 s, 0.53 s, and static (from left to right).

shown in Fig. 1(a). In terms of computational speed, the ESNN outperforms the DER method by roughly 60 times (Supplementary Information S10) as shown in Fig. 1(b). With the NN weights and all physical parameters fixed, we test the generalization of the ESNN on four previously unseen cases: on a Slinky (1) of a different number of cycles; (2) under a different boundary condition; (3) of a different density; (4) of a different Young's modulus (refer to Supplementary Information S9, Movie S4, and Movie S5 for (3) and (4)). In test case (1), the Slinky cycle number is changed to 40 with both ends clamped, as shown in Fig. 3(d). The Slinky is initially held horizontal and then dropped freely under gravity. The motion time shots of the Slinky from experiment, DER simulation, and ESNN simulation are compared in Fig. 3(d) and show an excellent agreement. In test case (2), the 40-cycle Slinky is held vertically, clamped at the upper end, and dropped freely from its undeformed configuration. The motion time shots from different methods are compared in Fig. 3(e) and again we observe a good agreement. In these two test cases, a satisfactory agreement is achieved without making any changes to the ESNN. The agreement originates from the embedded Euclidean symmetry and the fact that the NN is learning generalizable physics locally. As a contrast, [6,52] construct NN-based surrogate models for full-field solutions under a certain boundary condition. The benefit is that the NN prediction is extremely fast since only one forward-pass through the trained NN is required for each prediction. The price paid is in the generalization ability. For a new type of boundary condition, a new dedicated training dataset is required and the NN needs to learn from scratch. However, in our ESNN approach, different boundary conditions can be readily incorporated after training since the ESNN learns local physics which is boundary condition agnostic. In test case (2), the orientation of the Slinky is shifted by 90 degrees. The ESNN is still capable of generating the correct surrogate forces and system trajectory prediction since it preserves rotation equivariance.

3. Discussion

We demonstrate through the construction of alternate data-driven models why and how the three principal characteristics of our physics-aware ESNN model play crucial roles in ensuring accurate training based on data obtained from a *single demonstration* and then being able to generalize to unseen scenarios where *both initial and boundary conditions are varied*.

3.1. The importance of physical knowledge

To provide a physics-agnostic, data-driven baseline for comparison, a convolutional neural network (CNN) architecture is used to directly learn the one-step dynamics of the Slinky. Given the 2D coordinates of a Slinky at a time step, the CNN predicts the 2D Slinky coordinates at the next time step using a series of non-linear filters convoluting along the Slinky cycles with receptive field of three cycles. Such a convolutional architecture assumes homogeneity of the Slinky across cycles (as other methods do) and allows the application of the trained model to Slinkies consisting of arbitrary number of cycles. Boundary condition treatment is the same as that used in the ESNN (Supplementary Information S6). As shown in Fig. 4(b), the training result is fairly accurate. After training, the CNN is tested on a 40-cycle Slinky vertically held at the upper boundary. The unreasonable result in Fig. 4(c) shows that the CNN "memorizes" the data pattern in the training dataset, but fails to learn the underlying dynamics, therefore does not generalize to unseen conditions (Fig. 4(f)).

3.2. The importance of equivariance

Two other comparison methods are the force method and the energy method. The schematics of the methods are shown in Fig. 4(a). The NNs in the force method (referred as "force NN") and in the energy method (referred as "energy NN") share the same structure as the ESNN in terms of the number of layers and neurons per layer. See Materials and Methods section for training data preparation.

The force NN is trained to directly learn the mapping between the triplet coordinates and the elastic forces on the middle bars without equivariance encoded in the NN architecture. Although a reasonably low loss is achieved for training (Supplementary Information S11 and Fig. S9(a)), the force NN still fails to forward propagate using an ODE solver (Fig. 4(d)). Without the equivariance property, the force NN fails to generate physically reasonable and consistent predictions on unseen inputs. A natural thought to solve this issue is to train an enlarged NN on an augmented dataset with rotation and chiral transformations. This approach is unrealistic because the rotation angle is a continuous variable and the amount of data augmentation to enforce rotation-equivariance can be prohibitively huge. This highlights the necessity of incorporating physical symmetries to improve the efficacy and efficiency of NN-based reduced-order models.

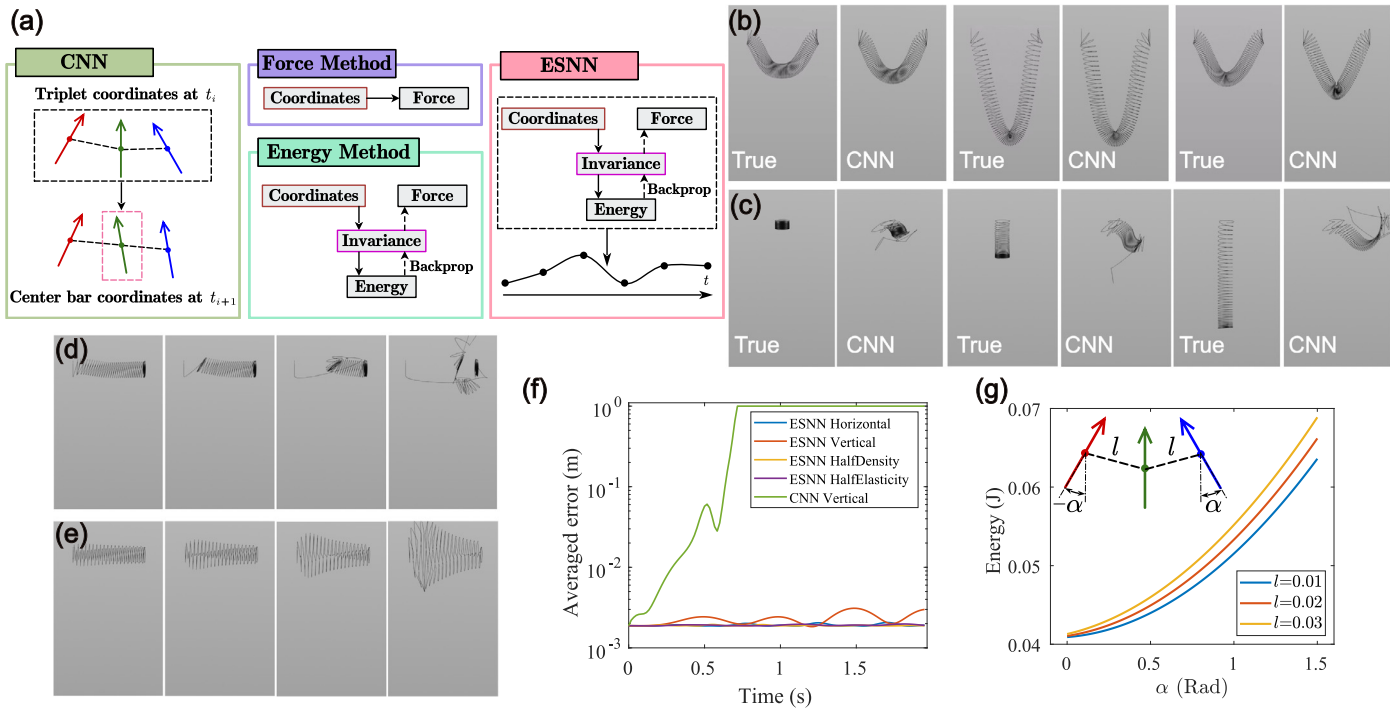

Fig. 4. (color online). Comparison between the ESNN and 3 other methods (CNN, force method, energy method). (a) Schematic comparison between different approaches. CNN takes the triplet coordinates at the current time step and predicts the center bar coordinates at the next time step without any physical symmetry or dynamics encoded. The force method has the same input but predicts the forces on the center bar. The energy method incorporates invariances to predict the energy first and then by backpropagation derives an equivariant force field. The force and energy methods perform data-fitting directly. The ESNN trains an invariant NN under the NODE framework. (b) Comparison between 3D DER ground truth and CNN training results (Supplementary Movie S6). Time shots are at 0.20 s, 0.43 s, and 1.18 s (from left to right). (c) Comparison between 3D DER ground truth and CNN generalization results (Supplementary Movie S7). Time shots are at 0.03 s, 0.17 s, and 0.33 s (from left to right). (d) The training results of the force method (Supplementary Movie S8) at 9.4 ms, 11.4 ms, 13.4 ms, and 14.9 ms (from left to right). The force method fails to generate a reasonable simulation. (e) The training results of the energy method (Supplementary Movie S9) at 7.5 ms, 8.5 ms, 9.5 ms, and 10.5 ms. The simulation diverges at 10.7 ms. (f) The error comparison between the ESNN (4 generalization cases) and CNN (generalization for a 40 cycle Slinky). The error is the vertex distance between NN prediction and 3D DER averaged on all the Slinky vertices. The CNN generalization error is significantly larger than those of the ESNN (capped at 1.0 for visualization). (g) The learnt energy of the ESNN at different bending angles α. The symmetric rotation configuration in the inlet ensures that at different bending angles the stretching stays the same and the shear remains 0. The energy increase is purely caused by bending motion. The bending energy and the associated bending moment are different for the same bending angle under different stretchings. This indicates a nonlinear coupling between stretching and bending.

3.3. The importance of the NODE training

The results from the energy NN are shown in Fig. 4(e). The energy NN is trained to predict the elastic energies given the triplet coordinates. Similar to the ESNN, the elastic force on the middle bar, which is the negative energy gradient w.r.t. the coordinates of the middle bar, can be accordingly derived with backpropagation. The energy NN differs from the ESNN in the training scheme: the energy NN is trained on coordinates-energy data pairs; the backpropagation to calculate forces is performed during the testing phase for forward simulation. Recall that for the ESNN, the backpropagation is already embedded in the training phase under the NODE framework to fit the Slinky trajectory along time. The energy NN simulation diverges at 10.7 ms. The divergence is due to two reasons: (1) Although it achieves a nearly perfect energy fitting (Supplementary Information S11 and Fig. S9(b)), there is no guarantee that the derivatives of energies, i.e. forces; (2) There will inevitably be errors in the energy prediction of the NN and the associated force calculation after training. The errors after propagating through an ODE solver should not diverge the simulation. This property is guaranteed when the NN is trained under the NODE framework, but not when directly fitting input–output data pairs. This highlights the necessity of using NODE as the training scheme for NN-based reduced-order models.

3.4. Comparison with classic 2D Slinky models

By investigating the prediction of the intermediate energy surrogate of the ESNN, we show that it automatically discovers the nonlinear stretching–bending coupling of the Slinky without any prior knowledge. Fig. 4(g) shows the triplet energy at different bending angles with 3 initial stretching lengths. As the initial stretching length increases, it takes more energy to bend the bars to a certain angle, i.e., the instantaneous bending stiffness is a function of the stretching length, which is a physical insight not reflected in classic models [42].

3.5. Other machine learning techniques for modeling physical systems

Physics-informed neural networks (PINN) [53,54]. PINN is typically provided the governing partial differential equations (PDEs) of a given system and then computes the solution. PINN can also solve inverse problems such as inferring unknown field variables and parameters from measured data, but the system model should typically be provided in a symbolic form. Thus, PINN is not suitable for our task, where we aim at constructing the system model from data without knowing the symbolic form.

Neural Operators [6,55–57]. Neural operator (NO) methods aim to learn "operators", which map a function to another function, from data, e.g., the mapping from an arbitrary initial condition as a function of spatial coordinates to the state of a PDE system

at a *fixed* time. NOs have demonstrated their effectiveness in multiple PDE examples. However, such methods have limitations preventing their application to the Slinky system. Tailored for general end-to-end learning of PDE systems, NOs are not able to incorporate the inductive bias of Newton's Second Law, which consequently also limits the incorporation of Euclidean symmetry. Further, due to the end-to-end characteristics, a single neural operator model is not applicable to Slinkies of different number of cycles.

Sparse Identification of Nonlinear Dynamics (SINDy) [22,58]. SINDy is a machine learning method that uses data obtained from observations of a physical system to construct analytical models of the system in the form of compact symbolic expressions chosen from a dictionary of predefined terms. Although SINDy has shown considerable promise, e.g., in model construction of systems with field variables such as vortex shedding behind an obstacle, it is not suitable for our problem. To see this, one simply needs to consider deriving the governing equations of a quadruple pendulum. It is not clear how a dictionary of combinatorial symbolic terms would be able to accurately and fully incorporate the complex rational terms involved in such a system. In other words, the curse of dimensionality would prevent SINDy from being applied to the Slinky system.

4. Conclusions

We have proposed an ESNN-based approach for building data-driven reduced-order models of physical systems under the NODE framework. We have validated its accuracy and extensive generalization ability, a critical differentiation from purely surrogate models, on real-world Slinky experiments. A roughly 60 times computational acceleration is achieved compared with classic simulation methods. The generalization ability originates from incorporating data-driven deep neural network models with physics principles including Euclidean symmetry and Newton's second law of motion. With these features, the ESNN is able to learn the reduced-order physics from a single demonstration case, and perform numerically stable, accurate, and fast predictions on a variety of unseen cases. We also show that the ESNN automatically discovers the phenomenon of nonlinear stretching–bending coupling that emerges in a Slinky without any prior knowledge. By incorporating only a universal geometric symmetry instead of any domain specific knowledge, the ESNN possesses the promise to be applicable to a wide variety of physical systems for automatic model construction and knowledge discovery.

CRediT authorship contribution statement

Qiaofeng Li: Methodology, Software, Validation, Visualization, Writing – original draft. **Tianyi Wang:** Methodology, Software, Writing – original draft. **Vwani Roychowdhury:** Conceptualization, Resources, Writing – review & editing, Supervision. **M. Khalid Jawed:** Conceptualization, Resources, Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgements and funding

We thank Mingjian Lu for his assistance on simulation implementation, and Zhuonan Hao and Dezhong Tong for their assistance on experiments. Following research grants are gratefully acknowledged: NSF (CMMI-2053971) for Q.L., T.W., V.R., and M.K.J.; NSF (IIS-1925360) for Q.L. and M.K.J.; and NSF (CAREER-2047663, CMMI-2101751) for M.K.J.

Appendix A. Discrete elastic rod simulation

Discrete Elastic Rod method simulates the dynamics of elastic rods under large nonlinear deformations. We implemented the Incremental Potential Contact model (Supplementary Information S2) to deal with the contact between Slinky cycles in both DER and NN-based simulations. For details, please refer to Supplementary Information S1.

Appendix B. Data generation for force and energy methods

After the 3D simulation is run, the total energy of each discrete elastic rod at each time step, including bending, stretching, and twisting energy, is recorded. The total energy of each Slinky cycle is the summation of the energies from the associated discrete rods. The 2D triplet coordinates and the corresponding total energy of the middle cycles constitute the inputs and outputs of the training data pairs for the energy method. The NN in the energy method is supposed to learn the mapping between triplet coordinates and middle bar energies.

After converting the 3D Slinky trajectory to 2D bar trajectory, the instantaneous acceleration of each bar is calculated with finite difference method. The pseudo elastic force on each bar is calculated by Newton's second law of motion. The NN in the force method is supposed to directly learn the mapping between triplet coordinates and middle bar pseudo forces.

Appendix C. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.eml.2022.101925.

References

[1] P. Benner, M. Ohlberger, A. Cohen, K. Willcox, Model Reduction and Approximation, in: Computational Science & Engineering, Society for Industrial and Applied Mathematics, 2017.
[2] D. Hartman, L.K. Mestha, A deep learning framework for model reduction of dynamical systems, in: 2017 IEEE Conference on Control Technology and Applications (CCTA), 2017, pp. 1917–1922.
[3] L. Fulton, V. Modi, D. Duvenaud, D.I.W. Levin, A. Jacobson, Latent-space dynamics for reduced deformable simulation, Comput. Graph. Forum 38 (2) (2019) 379–391.
[4] R. Maulik, A. Mohan, B. Lusch, S. Madireddy, P. Balaprakash, D. Livescu, Time-series learning of latent-space dynamics for reduced-order model closure, Physica D 405 (2020) 132368.
[5] T.P. Sapsis, A.J. Majda, Statistically accurate low-order models for uncertainty quantification in turbulent dynamical systems, Proc. Natl. Acad. Sci. 110 (34) (2013) 13705–13710.
[6] Z. Li, N.B. Kovachki, K. Azizzadenesheli, B. liu, K. Bhattacharya, A. Stuart, A. Anandkumar, Fourier neural operator for parametric partial differential equations, in: International Conference on Learning Representations, 2021.
[7] O. Semeraro, S. Bagheri, L. Brandt, D.S. Henningson, Feedback control of three-dimensional optimal disturbances using reduced-order models, J. Fluid Mech. 677 (2011) 63–102.
[8] O. Goury, C. Duriez, Fast, generic, and reliable control and simulation of soft robots using model order reduction, IEEE Trans. Robot. 34 (6) (2018) 1565–1576.
[9] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444.
[10] G.E. Karniadakis, I.G. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, L. Yang, Physics-informed machine learning, Nat. Rev. Phys. 3 (6) (2021) 422–440.

[11] G.X. Gu, C.-T. Chen, M.J. Buehler, De novo composite design based on machine learning algorithm, Extreme Mech. Lett. 18 (2018) 19–28.
[12] R. Fournier, L. Wang, O.V. Yazyev, Q. Wu, Artificial neural network approach to the analytic continuation problem, Phys. Rev. Lett. 124 (5) (2020) 056401.
[13] Z. Qin, L. Wu, H. Sun, S. Huo, T. Ma, E. Lim, P.-Y. Chen, B. Marelli, M.J. Buehler, Artificial intelligence method to design and fold alpha-helical structural proteins from the primary amino acid sequence, Extreme Mech. Lett. 36 (2020) 100652.
[14] E. Heiden, D. Millard, E. Coumans, Y. Sheng, G.S. Sukhatme, NeuralSim: Augmenting differentiable simulators with neural networks, in: 2021 IEEE International Conference on Robotics and Automation, ICRA, 2021, pp. 9474–9481.
[15] D. Kochkov, J.A. Smith, A. Alieva, Q. Wang, M.P. Brenner, S. Hoyer, Machine learning–accelerated computational fluid dynamics, Proc. Natl. Acad. Sci. 118 (21) (2021) e2101784118.
[16] C.-T. Chen, G.X. Gu, Learning hidden elasticity with deep neural networks, Proc. Natl. Acad. Sci. 118 (31) (2021) e2102721118.
[17] S. Ye, W.-Z. Huang, M. Li, X.-Q. Feng, Deep learning method for determining the surface elastic moduli of microstructured solids, Extreme Mech. Lett. 44 (2021) 101226.
[18] S. Yin, Z. Jia, X. Li, J. Zhu, Y. Xu, T. Li, Machine-learning-accelerated design of functional structural components in deep-sea soft robots, Extreme Mech. Lett. 52 (2022) 101635.
[19] F.Y. Liu, B. Ni, M.J. Buehler, PRESTO: Rapid protein mechanical strength prediction with an end-to-end deep learning model, Extreme Mech. Lett. 55 (2022) 101803.
[20] A.J.K. Chua, C.R. Galley, M. Vallisneri, Reduced-order modeling with artificial neurons for gravitational-wave inference, Phys. Rev. Lett. (21) (2019) 211101.
[21] Y. Rubanova, R.T.Q. Chen, D.K. Duvenaud, Latent ODEs for irregularly-sampled time series, in: Advances in Neural Information Processing Systems, Vol. 32, 2019.
[22] K. Champion, B. Lusch, J.N. Kutz, S.L. Brunton, Data-driven discovery of coordinates and governing equations, Proc. Natl. Acad. Sci. 116 (45) (2019) 22445–22451.
[23] Y. Bar-Sinai, S. Hoyer, J. Hickey, M.P. Brenner, Learning data-driven discretizations for partial differential equations, Proc. Natl. Acad. Sci. USA 116 (31) (2019) 15344–15349.
[24] R. Iten, T. Metger, H. Wilming, L.d. Rio, R. Renner, Discovering physical concepts with neural networks, Phys. Rev. Lett. 124 (1) (2020) 010508.
[25] J. Zeng, L. Cao, M. Xu, T. Zhu, J.Z.H. Zhang, Complex reaction processes in combustion unraveled by neural network-based molecular dynamics simulation, Nature Commun. 11 (1) (2020) 5713.
[26] Z. Liu, M. Tegmark, Machine learning conservation laws from trajectories, Phys. Rev. Lett. 126 (18) (2021) 180604.
[27] P. Zhang, H. Shen, H. Zhai, Machine learning topological invariants with neural networks, Phys. Rev. Lett. 120 (6) (2018) 066401.
[28] K. Atz, F. Grisoni, G. Schneider, Geometric deep learning on molecular representations, Nat. Mach. Intell. 3 (12) (2021) 1023–1032.
[29] S. Greydanus, M. Dzamba, J. Yosinski, Hamiltonian neural networks, in: Advances in Neural Information Processing Systems, Vol. 32, 2019.
[30] M. Cranmer, S. Greydanus, S. Hoyer, P. Battaglia, D. Spergel, S. Ho, Lagrangian neural networks, in: ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations, 2020.
[31] M.M. Bronstein, J. Bruna, T. Cohen, P. Veličković, Geometric deep learning: Grids, groups, graphs, geodesics, and gauges, 2021, Preprint at https://arxiv.org/abs/2104.13478.
[32] M. Weiler, M. Geiger, M. Welling, W. Boomsma, T. Cohen, 3D steerable CNNs: learning rotationally equivariant features in volumetric data, in: Proceedings of the 32nd International Conference on Neural Information Processing Systems, 2018, pp. 10402–10413.
[33] N. Thomas, T. Smidt, S. Kearnes, L. Yang, L. Li, K. Kohlhoff, P. Riley, Tensor field networks: Rotation- and translation-equivariant neural networks for 3D point clouds, 2018, Preprint at https://arxiv.org/abs/1802.08219.
[34] M. Finzi, M. Welling, A.G. Wilson, A practical method for constructing equivariant multilayer perceptrons for arbitrary matrix groups, 2021, Preprint at https://arxiv.org/abs/2104.09459.
[35] V.G. Satorras, E. Hoogeboom, M. Welling, E(n) equivariant graph neural networks, 2021, Preprint at https://arxiv.org/abs/2102.09844.
[36] S. Batzner, A. Musaelian, L. Sun, M. Geiger, J.P. Mailoa, M. Kornbluth, N. Molinari, T.E. Smidt, B. Kozinsky, E(3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials, Nature Commun. 13 (1) (2022) 2453.
[37] E. Gawehn, J.A. Hiss, G. Schneider, Deep learning in drug discovery, Mol. Inform. 35 (1) (2016) 3–14.
[38] J. Jiménez-Luna, F. Grisoni, N. Weskamp, G. Schneider, Artificial intelligence in drug discovery: recent advances and future perspectives, Expert Opin. Drug Discovery 16 (9) (2021) 1–11.
[39] J. Gilmer, S.S. Schoenholz, P.F. Riley, O. Vinyals, G.E. Dahl, Neural message passing for quantum chemistry, in: Proceedings of the 34th International Conference on Machine Learning - Volume 70, in: ICML'17, JMLR.org, Sydney, NSW, Australia, 2017, pp. 1263–1272.
[40] P.T. Komiske, E.M. Metodiev, J. Thaler, Energy Flow Networks: Deep Sets for particle jets, J. High Energy Phys. 2019 (1) (2019).
[41] H. Qu, L. Gouskos, ParticleNet: Jet tagging via particle clouds, Phys. Rev. D 101 (5) (2020) 056019.
[42] D.P. Holmes, A.D. Borum, B.F. Moore, R.H. Plaut, D.A. Dillard, Equilibria and instabilities of a Slinky: Discrete model, Int. J. Non-Linear Mech. 65 (2014) 236–244.
[43] R.T.Q. Chen, Y. Rubanova, J. Bettencourt, D.K. Duvenaud, Neural ordinary differential equations, in: Advances in Neural Information Processing Systems, Vol. 31, 2018.
[44] M. Bergou, M. Wardetzky, S. Robinson, B. Audoly, E. Grinspun, Discrete elastic rods, ACM Trans. Graph. (SIGGRAPH) 27 (3) (2008) 63:1–63:12.
[45] G. Huang, Z. Liu, L. Van Der Maaten, K.Q. Weinberger, Densely connected convolutional networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 4700–4708.
[46] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, PyTorch: An imperative style, high-performance deep learning library, in: Advances in Neural Information Processing Systems 32, 2019, pp. 8024–8035.
[47] M.K. Jawed, F. Da, J. Joo, E. Grinspun, P.M. Reis, Coiling of elastic rods on rigid substrates, Proc. Natl. Acad. Sci. 111 (41) (2014) 14663–14668.
[48] M.K. Jawed, N.K. Khouri, F. Da, E. Grinspun, P.M. Reis, Propulsion and instability of a flexible helical rod rotating in a viscous fluid, Phys. Rev. Lett. 115 (16) (2015) 168101.
[49] W. Huang, X. Huang, C. Majidi, M.K. Jawed, Dynamic simulation of articulated soft robots, Nature Commun. 11 (1) (2020) 2233.
[50] W. Huang, Y. Wang, X. Li, M.K. Jawed, Shear induced supercritical pitchfork bifurcation of pre-buckled bands, from narrow strips to wide plates, J. Mech. Phys. Solids 145 (2020) 104168.
[51] M. Li, Z. Ferguson, T. Schneider, T. Langlois, D. Zorin, D. Panozzo, C. Jiang, D.M. Kaufman, Incremental potential contact: Intersection- and inversion-free large deformation dynamics, ACM Trans. Graph. (SIGGRAPH) 39 (4) (2020).
[52] E. Haghighat, M. Raissi, A. Moure, H. Gomez, R. Juanes, A physics-informed deep learning framework for inversion and surrogate modeling in solid mechanics, Comput. Methods Appl. Mech. Engrg. 379 (2021) 113741.
[53] M. Raissi, P. Perdikaris, G. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, J. Comput. Phys. 378 (2019) 686–707.
[54] L. Yang, X. Meng, G.E. Karniadakis, B-PINNs: Bayesian physics-informed neural networks for forward and inverse PDE problems with noisy data, J. Comput. Phys. 425 (2021) 109913.
[55] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, A. Anandkumar, Neural operator: Graph kernel network for partial differential equations, 2020, Preprint at https://arxiv.org/abs/2003.03485.
[56] Z. Li, H. Zheng, N. Kovachki, D. Jin, H. Chen, B. Liu, K. Azizzadenesheli, A. Anandkumar, Physics-informed neural operator for learning partial differential equations, 2021, Preprint at https://arxiv.org/abs/2111.03794,
[57] L. Lu, P. Jin, G. Pang, Z. Zhang, G.E. Karniadakis, Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators, Nat. Mach. Intell. 3 (3) (2021) 218–229.
[58] S.L. Brunton, J.L. Proctor, J.N. Kutz, Discovering governing equations from data by sparse identification of nonlinear dynamical systems, Proc. Natl. Acad. Sci. 113 (15) (2016) 3932–3937.