Noether Networks: Meta-Learning Useful Conserved Quantities

Ferran Alet[*1], **Dylan Doblar**[*1], **Allan Zhou**[2],
Joshua Tenenbaum[1], **Kenji Kawaguchi**[3], **Chelsea Finn**[2]
[1]MIT, [2]Stanford University, [3]National University of Singapore
{alet,ddoblar}@mit.edu

Abstract

Progress in machine learning (ML) stems from a combination of data availability, computational resources, and an appropriate encoding of inductive biases. Useful biases often exploit symmetries in the prediction problem, such as convolutional networks relying on translation equivariance. Automatically discovering these useful symmetries holds the potential to greatly improve the performance of ML systems, but still remains a challenge. In this work, we focus on sequential prediction problems and take inspiration from Noether's theorem to reduce the problem of finding inductive biases to meta-learning useful conserved quantities. We propose Noether Networks: a new type of architecture where a meta-learned conservation loss is optimized inside the prediction function. We show, theoretically and experimentally, that Noether Networks improve prediction quality, providing a general framework for discovering inductive biases in sequential problems.

1 Introduction

The clever use of inductive biases to exploit symmetries has been at the heart of many landmark achievements in machine learning, such as translation invariance in CNN image classification [25], permutation invariance in Graph Neural Networks [34] for drug design [38], and roto-translational equivariance in SE3-transformers [19] for protein structure prediction. However, there may be exploitable symmetries that are either unknown or difficult to describe with code. In this work, we aim to find symmetries in sequential prediction problems, inspired by Noether's theorem [29], which loosely states the following:

For every continuous symmetry property of a dynamical system,
there is a corresponding quantity whose value is conserved in time.

Motivated by this equivalence, we propose that conservation laws are a powerful paradigm for meta-learning useful inductive biases in sequential problems. Whereas symmetries are global properties linked to counter-factuals about perturbing the data, conserved quantities can be evaluated using only the true data. This provides an immediate signal for machine learning algorithms to exploit.

Our approach involves meta-learning a parametric conservation loss function which is useful to the prediction task. We leverage the *tailoring* framework [3], which proposes to encode inductive biases by fine-tuning neural networks with hand-designed unsupervised losses inside the prediction function. In contrast to traditional auxiliary losses, which are added to the task loss during training, tailoring optimizes both during training and testing, customizing the model to each individual query to ensure there is no generalization gap for the self-supervised loss. We propose to meta-learn the self-supervised loss function and parameterize it in the form of a conservation loss over time

[*]Equal contribution. Our code is publicly available at https://lis.csail.mit.edu/noether.

Fourth Workshop on Machine Learning and the Physical Sciences (NeurIPS 2021).

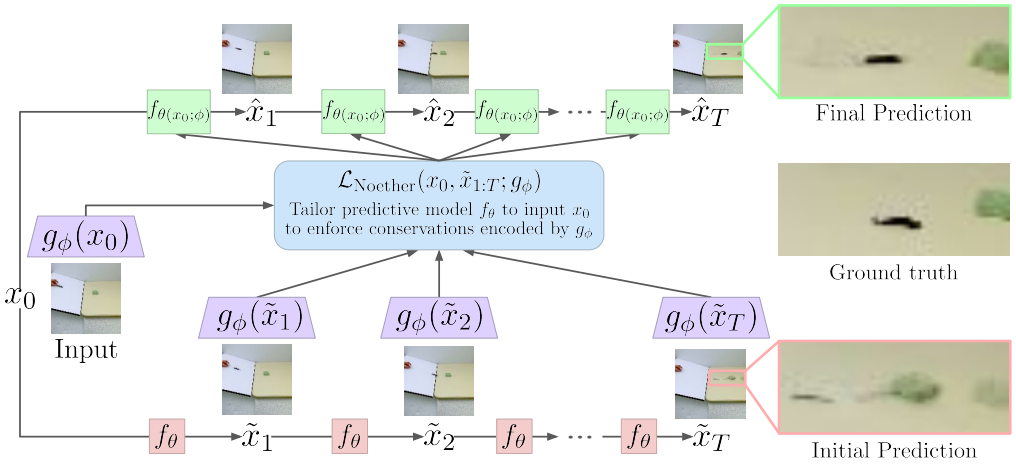

Figure 1: Noether Networks enforce conservation laws, meta-learned by g_ϕ, in sequential predictions made by f_θ, which is tailored to the input x_0 to produce final predictions $\hat{x}_{1:T}$. Imposing these meta-learned inductive biases improves video prediction quality with objects sliding down a ramp.

steps t; i.e. $\mathcal{L}(x_0, \tilde{x}_{1:T}, \phi) = \sum_{t=1}^{T} |g_\phi(x_0) - g_\phi(\tilde{x}_t)|^2$. By doing so, we allow the network to discover its own inductive biases. The conservation form encodes a meta-inductive bias (inductive bias over inductive biases) which narrows the search space exponentially (in prediction horizon T) and simplifies the parameterization. We show experimentally how this framework allows us to recover known conservations and use them for better generalization in scientific data, as well as learn useful conservation laws from raw pixels.

2 Theoretical advantages of enforcing conservation laws

In this section, we demonstrate principled advantages of enforcing conservation laws of the form $g_\phi(f_\theta(x)) = g_\phi(x)$ by considering a special case where preimages under g_ϕ form affine subspaces.

Let input x and target y be $x, y \in \mathbb{R}^d$, and let the Noether embedding be $g_\phi : \mathbb{R}^d \to \mathcal{P}$ where $\mathcal{P} = \{g_\phi(x) : x \in \mathbb{R}^d\}$. We consider a restricted class of models, parameterized by $\theta \in \Theta$, of the form $f_\theta(x) = x + v_\theta$ for $v_\theta \in \mathbb{R}^d$ such that for all x, the preimage of g_ϕ is $g_\phi^{-1}[\{g_\phi(x)\}] = \{x + Az : z \in \mathbb{R}^m\}$, $A \in \mathbb{R}^{d \times m}$. Here, $m \leq d$ is the dimensionality of the preimages of g_ϕ. We denote by C the smallest upper bound on the loss value as $\mathcal{L}(f_\theta(x), y) \leq C$ (for all x, y and θ). Define $\psi(v) = \mathbb{E}_{x,y}[\mathcal{L}(f(x, v), y)] - \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(f(x_i, v), y_i)$, with Lipschitz constant ζ. Therefore, $\zeta \geq 0$ is the smallest real number such that, for all v and v' in $\mathcal{V}$, $|\psi(v) - \psi(v')| \leq \zeta \|v - v'\|_2$, where $\mathcal{V} = \{v_\theta \in \mathbb{R}^d : \theta \in \Theta\}$. Finally, we define $R = \sup_{\theta \in \Theta} \|v_\theta\|_2$, and the generalization gap by $G(\theta) = \mathbb{E}_{x,y}[\mathcal{L}(f_\theta(x), y)] - \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(f_\theta(x_i), y_i)$. Theorem 1 shows that enforcing conservation laws of $g_\phi(f_\theta(x)) = g_\phi(x)$ is advantageous when the dimension of the preimages under g_ϕ is less than the dimension of x; that is, when $m < d$.

Theorem 1. *Let $\rho \in \mathbb{N}^+$. Then, for any $\delta > 0$, with probability at least $1 - \delta$ over an iid draw of n examples $((x_i, y_i))_{i=1}^n$, the following holds for all $\theta \in \Theta$:*

$$G(\theta) \leq C\sqrt{\frac{\xi \ln(\max(\sqrt{\xi}, 1)) + \xi \ln(2R(\zeta^{1-1/\rho})\sqrt{n}) + \ln(1/\delta)}{2n}} + \mathbb{1}\{\xi \geq 1\}\sqrt{\frac{\zeta^{2/\rho}}{n}}. \quad (1)$$

where $\xi = m$ if $g_\phi(f_\theta(x)) = g_\phi(x)$ for any $x \in \mathbb{R}^d$ and $\theta \in \Theta$, and $\xi = d$ otherwise.

The proof is presented in Appendix A. In Theorem 1, when we enforce the conservation laws, d is replaced by m, the dimension of the preimage. So, enforcing conservations improves the bound on generalization error for $m < d$.

Algorithm 1 Prediction and training procedures for Noether Networks with neural conservation loss

Given: predictive model class f; embedding model class g; prediction horizon T; training dist. $\mathcal{D}_{\text{train}}$; batch size N; learning rates λ_{in}, λ_{out}, λ_{emb}; task loss $\mathcal{L}_{\text{task}}$; Noether loss $\mathcal{L}_{\text{Noether}}$

procedure PREDICTSEQUENCE($x_0; \theta, \phi$)
  $\tilde{x}_0, \hat{x}_0 \leftarrow x_0, x_0$
  $\tilde{x}_t \leftarrow f_\theta(\tilde{x}_{t-1})\ \forall t \in \{1, \dots, T\}$ ▷ Initial predictions
  $\theta(x_0; \phi) \leftarrow \theta - \lambda_{\text{in}} \nabla_\theta \mathcal{L}_{\text{Noether}}(x_0, \tilde{x}_{1:T}; g_\phi)$ ▷ Inner step with Noether loss
  $\hat{x}_t \leftarrow f_{\theta(x_0;\phi)}(\hat{x}_{t-1})\ \forall t \in \{1, \dots, T\}$ ▷ Final prediction with tailored weights
  return $\hat{x}_{1:T}$
procedure TRAIN
  $\phi \leftarrow$ randomly initialized weights ▷ Initialize weights for Noether embedding g
  $\theta \leftarrow$ randomly initialized weights ▷ Initialize weights for predictive model f
  while not done **do**
    Sample batch $x^{(0)}_{0:T}, \dots, x^{(N)}_{0:T} \sim \mathcal{D}_{\text{train}}$
    for $0 \le n \le N$ **do**
      $\hat{x}^{(n)}_{1:T} \leftarrow$ PREDICTSEQUENCE($x^{(n)}_0; \theta, \phi$)
    $\phi \leftarrow \phi - \lambda_{\text{emb}} \nabla_\phi \sum_{n=0}^{N} \mathcal{L}_{\text{task}}(\hat{x}^{(n)}_{1:T}, x^{(n)}_{1:T})$ ▷ Outer step with task loss for embedding
    $\theta \leftarrow \theta - \lambda_{\text{out}} \nabla_\theta \sum_{n=0}^{N} \mathcal{L}_{\text{task}}(\hat{x}^{(n)}_{1:T}, x^{(n)}_{1:T})$ ▷ Outer step for predictive model
  return ϕ, θ

3 Noether Networks

Leveraging tailoring to encode inductive biases. We perform a prediction-time optimization to encourage outputs to follow conservation laws using the tailoring and meta-tailoring frameworks [3]. Tailoring encodes inductive biases in the form of unsupervised losses optimized inside the prediction function. In doing so, tailoring fine-tunes the model to each query to ensure that it satisfies the unsupervised loss for that query. For example, we may optimize for energy conservation in a physics prediction problem. In meta-tailoring, we train the model to do well on the task loss after the tailoring step has fine-tuned its parameters. This allows us to impose inductive biases on the test queries not known at training time, as in transductive learning [46]. A limitation of tailoring is that the tailoring loss must be user-specified. This is acceptable in domains where the desired inductive bias is both known and easily encoded, but problematic in general — we address this issue with Noether Networks.

Meta-learning a neural loss function. We propose Noether Networks, an architecture class for sequential prediction that consists of a base predictor $f_\theta : \tilde{x}_t \mapsto \tilde{x}_{t+1}$ and a meta-learned tailoring loss, parameterized as the conservation of a neural embedding $g_\phi : \tilde{x}_t \mapsto \mathbb{R}^k$. This embedding takes raw predictions as input (such as images in the case of video prediction). The conservation form induces a meta-inductive bias over potential learned tailoring losses. The Noether loss is formulated as

$$\mathcal{L}_{\text{Noether}}(x_0, \tilde{x}_{1:T}; g_\phi) = \underbrace{\sum_{t=1}^{T} \left| g_\phi(x_0) - g_\phi\left(\tilde{x}_t\right) \right|^2}_{\text{(a)}} \approx \underbrace{\sum_{t=1}^{T} \left| g_\phi(\tilde{x}_{t-1}) - g_\phi\left(\tilde{x}_t\right) \right|^2}_{\text{(b)}} \tag{2}$$

where x_0 is the ground-truth input, the $\tilde{x}_t = f_\theta(\tilde{x}_{t-1})$ are the model's predictions, and $\tilde{x}_0 \triangleq x_0$. Expressions 2(a) and 2(b) are equivalent if we fully enforce the conservation law, but they differ if conservation is not fully enforced. When not fully conserving, 2(a) propagates information from ground truth more directly to the prediction, but 2(b) may be a useful approximation which better handles imperfectly conserved quantities, where the quantity should be Lipschitz but not exactly constant. In both cases, if we tailor θ with a single gradient step; the gradient update takes the form

$$\theta(x_0; \phi) = \theta - \lambda_{\text{in}} \nabla_\theta \mathcal{L}_{\text{Noether}}(x_0, \tilde{x}_{1:T}(\theta); g_\phi). \tag{3}$$

We compute final predictions as $\hat{x}_t = f_{\theta(x_0;\phi)}(\hat{x}_{t-1})$. We can now backpropagate from $\mathcal{L}_{\text{task}}(x_{1:T}, \hat{x}_{1:T})$ back to ϕ, which will be optimized so that the unsupervised adaptation $\theta \mapsto \theta(x_0; \phi)$ helps lower $\mathcal{L}_{\text{task}}$.The optimization requires second-order gradients to express how ϕ affects $\mathcal{L}_{\text{task}}$ through $\theta(x_0; \phi)$. This is similar to MAML [17], as well as works on meta-learning loss functions for few-shot learning [4] and group distribution shift [52]. Algorithm 1 provides pseudo-code.

Table 1: Recovering $\mathcal{H}$ for the ideal pendulum.

Method	Description	RMSE
Vanilla MLP	N/A	0.0563
Noether Nets	$p^2 - 2.99\cos(q)$	0.0423
True $\mathcal{H}$ [oracle]	$p^2 - 3.00\cos(q)$	0.0422

Table 2: Recovering $\mathcal{H}$ for the ideal spring.

Method	Description	RMSE
Vanilla MLP	N/A	.0174
Noether Nets	$q^2 + 1.002\,p^2$	.0165
True $\mathcal{H}$ [oracle]	$q^2 + 1.000\,p^2$	.0166

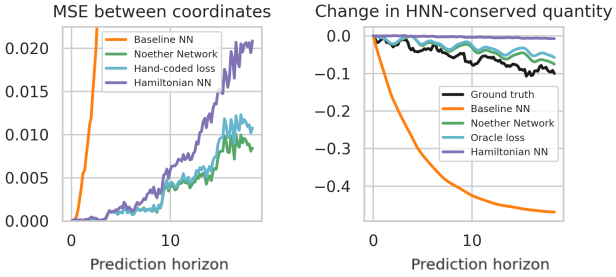

Figure 2: Noether Networks can recover the energy of a real pendulum, even though it is not fully conserved. This is because they only look for quantities whose conservation helps improve predictions. Moreover, by only softly encouraging conservation, it better encodes imperfect conservations.

4 Experiments

Can Noether Networks recover known conservation laws in scientific data? We first evaluate the capabilities of Noether networks on a set of symbolic regression tasks involving simulated and real physical systems. The ideal spring and ideal pendulum settings of Greydanus et al. [21] allow us to examine the behavior of Noether Networks for scientific data where we know a useful conserved quantity: the energy. We use their MLP baseline and discover conservation laws that, when used for meta-tailoring, can improve prediction performance. Since the baseline does not predict x_{t+1} but rather $\frac{d}{dt}(x_t)$, we apply the loss to a finite-difference approximation, i.e. $\mathcal{L}_{\text{Noether}}\left(x_t, x_t + f_\theta(x_t)\Delta_t\right)$.

For our pipelined approach, we program a simple domain specific language (DSL) for physical formulas. In contrast to most program synthesis approaches, our setting has continuous parameters which can be trained via gradient descent. We first search all valid symbolic formulas up to depth 7 and then train their parameters on the true data to be approximately conserved by minimizing variance of the formula on the training set using SGD. Then, we verify whether the formula indeed is approximately conserved by comparing its variance on true vs. random data.Finally, we try using each conserved expression as a Noether loss in the meta-tailoring setting (trying several learning rates for each), starting from the baseline MLP weights. The formula with the best performance on the training data is selected and evaluated on the test set. See Appendix C for additional experimental details.

For the ideal pendulum and spring, our method recovers the true Hamiltonian $\mathcal{H}$ almost perfectly. Using these expressions as losses reaches equivalent performance with that of the oracle, which fine-tunes using the true formula. Results are shown in Tables 1 and 2.

Finally, we run the same process for data coming from a real pendulum [35], where the Hamiltonian is not conserved. We use largely the same pipeline as for the ideal pendulum, the differences are explained in Appendix C. Noether Networks end up discovering $p^2 - 2.39\cos(q)$, compared to the (potentially sub-optimal) $\mathcal{H} = p^2 - 2.4\cos(q)$ described in [21]. Using the discovered loss improves the baseline MLP by more than one order of magnitude, matches the performance of hand-coding the conservation loss, and improves over Hamiltonian Neural Networks, which fully impose the conservation of energy. Results can be seen in Figure 2.

Can Noether Networks parameterize useful conserved quantities from raw pixel information? To characterize the benefits provided by Noether Networks in a real-world video prediction setting, we compare a Noether Network version of the SVG video prediction model — a variational autoencoder-based method [13] — to a standard SVG baseline on the 20 degree ramp scenario of the Physics 101 dataset [50]. The dataset contains videos of various real-world objects sliding down an incline and colliding with a second object. Each model is conditioned on the first two frames of a given sequence and must predict the subsequent 20 frames.

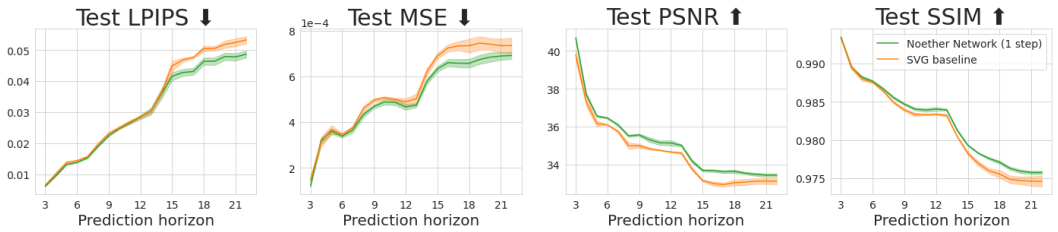

Figure 3: The Noether Network outperforms the baseline by a small margin in all four metrics in real-world video prediction, showing they can meta-learn useful conserved quantities from raw pixel data.

In our experiments, we train on only 311 training sequences, which makes generalization to the test set difficult. In this low-data setting, the baseline SVG model struggles with overfitting — predicted frames often morph objects in the test set into objects seen at train-time, and the object in motion often morphs with the target object (see the prediction $\tilde{x}_T$ in Figure 1, for example). Our Noether Network uses the inner loss formulation of Equation 2(b), where g_ϕ is parameterized as a two-layer CNN with a final fully connected projection layer that produces 64-dimensional embeddings. We meta-tailor the embedding and the model from scratch for 400 epochs. As seen in Figure 3, taking a single inner meta-tailoring step improves performance marginally over the baseline SVG model on the test set w.r.t. MSE, PSNR, and SSIM. These results provide some evidence that Noether Networks are capable of learning useful inductive biases from raw video data.

5 Conclusion

We propose Noether Networks: a new framework for meta-learning inductive biases through approximately conserved quantities imposed at prediction time. Originally motivated by applications in physics, our results indicate that Noether networks can recover known conservation laws in scientific data and also provide modest gains when applied to video prediction from raw pixel data. The generality of optimizing arbitrary unsupervised losses at prediction time provides an initial step towards broader applications. Finally, this work points at the usefulness of designing meta-learned inductive biases by putting priors on the biases instead of hard-coding biases directly.

Acknowledgements

We want to thank Leslie Kaelbling, Tomás Lozano-Pérez and Maria Bauza for insightful feedback. We gratefully acknowledge support from GoodAI; from NSF grant 1723381; from AFOSR grant FA9550-17-1-0165; from ONR grants N00014-18-1-2847 and N00014-21-1-2685; from the Honda Research Institute, from MIT-IBM Watson Lab; from SUTD Temasek Laboratories; and from Google. Chelsea Finn is a fellow in the CIFAR Learning in Machines and Brains program. We also acknowledge the MIT SuperCloud and Lincoln Laboratory Supercomputing Center for providing HPC resources that have contributed to the reported research results. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of our sponsors.

References

[1] F. Alet, A. K. Jeewajee, M. Bauza, A. Rodriguez, T. Lozano-Perez, and L. P. Kaelbling. Graph element networks: adaptive, structured computation and memory. *Proceedings of the 36th International Conference on Machine Learning-Volume 97*, 2019.

[2] F. Alet, E. Weng, T. Lozano-Perez, and L. Kaelbling. Neural relational inference with fast modular meta-learning. In *Advances in Neural Information Processing Systems (NeurIPS) 32*, 2019.

[3] F. Alet, M. Bauza, K. Kawaguchi, N. G. Kuru, T. Lozano-Perez, and L. P. Kaelbling. Tailoring: encoding inductive biases by optimizing unsupervised objectives at prediction time. *arXiv preprint arXiv:2009.10623*, 2020.

[4] A. Antoniou and A. J. Storkey. Learning to learn by self-critique. In *Advances in Neural Information Processing Systems*, pages 9940–9950, 2019.

[5] P. W. Battaglia, R. Pascanu, M. Lai, D. Rezende, and K. Kavukcuoglu. Interaction networks for learning about objects, relations and physics. *arXiv preprint arXiv:1612.00222*, 2016.

[6] G. Benton, M. Finzi, P. Izmailov, and A. G. Wilson. Learning invariances in neural networks. *arXiv preprint arXiv:2010.11882*, 2020.

[7] M. M. Bronstein, J. Bruna, T. Cohen, and P. Veličković. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

[8] M. B. Chang, T. Ullman, A. Torralba, and J. B. Tenenbaum. A compositional object-based approach to learning physical dynamics. *arXiv preprint arXiv:1612.00341*, 2016.

[9] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[10] M. Cranmer, S. Greydanus, S. Hoyer, P. Battaglia, D. Spergel, and S. Ho. Lagrangian neural networks. *arXiv preprint arXiv:2003.04630*, 2020.

[11] M. Cranmer, A. Sanchez-Gonzalez, P. Battaglia, R. Xu, K. Cranmer, D. Spergel, and S. Ho. Discovering symbolic models from deep learning with inductive biases. *arXiv preprint arXiv:2006.11287*, 2020.

[12] F. de Avila Belbute-Peres, T. D. Economon, and J. Zico Kolter. Combining differentiable pde solvers and graph neural networks for fluid flow prediction. *arXiv e-prints*, pages arXiv–2007, 2020.

[13] E. Denton and R. Fergus. Stochastic video generation with a learned prior. In *International Conference on Machine Learning*, pages 1174–1183. PMLR, 2018.

[14] S. A. Desai, M. Mattheakis, D. Sondak, P. Protopapas, and S. J. Roberts. Port-hamiltonian neural networks for learning explicit time-dependent dynamical systems. *Physical Review E*, 104(3):034312, 2021.

[15] K. Ellis, C. Wong, M. Nye, M. Sable-Meyer, L. Cary, L. Morales, L. Hewitt, A. Solar-Lezama, and J. B. Tenenbaum. Dreamcoder: Growing generalizable, interpretable knowledge with wake-sleep bayesian program learning. *arXiv preprint arXiv:2006.08381*, 2020.

[16] J. L. Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

[17] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*, 2017.

[18] M. Finzi, K. A. Wang, and A. G. Wilson. Simplifying hamiltonian and lagrangian neural networks via explicit constraints. *arXiv preprint arXiv:2010.13581*, 2020.

[19] F. B. Fuchs, D. E. Worrall, V. Fischer, and M. Welling. Se (3)-transformers: 3d roto-translation equivariant attention networks. *arXiv preprint arXiv:2006.10503*, 2020.

[20] G. Głuch and R. Urbanke. Noether: The more things change, the more stay the same. *arXiv preprint arXiv:2104.05508*, 2021.

[21] S. Greydanus, M. Dzamba, and J. Yosinski. Hamiltonian neural networks. In *Advances in Neural Information Processing Systems*, pages 15353–15363, 2019.

[22] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

[23] M. I. Jordan. Serial order: A parallel distributed processing approach. In *Advances in psychology*, volume 121, pages 471–495. Elsevier, 1997.

[24] T. Kipf, E. Fetaya, K.-C. Wang, M. Welling, and R. Zemel. Neural relational inference for interacting systems. *arXiv preprint arXiv:1802.04687*, 2018.

[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[26] Z. Liu and M. Tegmark. Ai poincaré: Machine learning conservation laws from trajectories. *arXiv preprint arXiv:2011.04698*, 2020.

[27] M. Lutter, C. Ritter, and J. Peters. Deep lagrangian networks: Using physics as model prior for deep learning. *arXiv preprint arXiv:1907.04490*, 2019.

[28] L. Metz, N. Maheswaranathan, B. Cheung, and J. Sohl-Dickstein. Meta-learning update rules for unsupervised representation learning. *arXiv preprint arXiv:1804.00222*, 2018.

[29] E. Noether. Invariante variationsprobleme. *Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen, Mathematisch-Physikalische Klasse*, 1918:235–257, 1918.

[30] T. Pfaff, M. Fortunato, A. Sanchez-Gonzalez, and P. W. Battaglia. Learning mesh-based simulation with graph networks. *arXiv preprint arXiv:2010.03409*, 2020.

[31] E. Rossi, B. Chamberlain, F. Frasca, D. Eynard, F. Monti, and M. Bronstein. Temporal graph networks for deep learning on dynamic graphs. *arXiv preprint arXiv:2006.10637*, 2020.

[32] A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, and P. Battaglia. Learning to simulate complex physics with graph networks. In *International Conference on Machine Learning*, pages 8459–8468. PMLR, 2020.

[33] V. G. Satorras, E. Hoogeboom, and M. Welling. E (n) equivariant graph neural networks. *arXiv preprint arXiv:2102.09844*, 2021.

[34] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

[35] M. Schmidt and H. Lipson. Distilling free-form natural laws from experimental data. *science*, 324(5923):81–85, 2009.

[36] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[37] K. O. Stanley, D. B. D'Ambrosio, and J. Gauci. A hypercube-based encoding for evolving large-scale neural networks. *Artificial life*, 15(2):185–212, 2009.

[38] J. M. Stokes, K. Yang, K. Swanson, W. Jin, A. Cubillos-Ruiz, N. M. Donghia, C. R. MacNair, S. French, L. A. Carfrae, Z. Bloom-Ackermann, et al. A deep learning approach to antibiotic discovery. *Cell*, 180(4):688–702, 2020.

[39] H. Suh and R. Tedrake. The surprising effectiveness of linear models for visual foresight in object pile manipulation. *arXiv preprint arXiv:2002.09093*, 2020.

[40] Y. Sun, X. Wang, Z. Liu, J. Miller, A. A. Efros, and M. Hardt. Test-time training for out-of-distribution generalization. *arXiv preprint arXiv:1909.13231*, 2019.

[41] C. Tallec and Y. Ollivier. Can recurrent neural networks warp time? *arXiv preprint arXiv:1804.11188*, 2018.

[42] H. Tanaka and D. Kunin. Noether's learning dynamics: The role of kinetic symmetry breaking in deep learning. *arXiv preprint arXiv:2105.02716*, 2021.

[43] 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 preprint arXiv:1802.08219*, 2018.

[44] S.-M. Udrescu and M. Tegmark. Ai feynman: A physics-inspired method for symbolic regression. *Science Advances*, 6(16):eaay2631, 2020.

[45] S.-M. Udrescu and M. Tegmark. Symbolic pregression: Discovering physical laws from distorted video. *Physical Review E*, 103(4):043307, 2021.

[46] V. Vapnik. *The nature of statistical learning theory*. Springer science & business media, 1995.

[47] D. Wang, E. Shelhamer, S. Liu, B. Olshausen, and T. Darrell. Fully test-time adaptation by entropy minimization. *arXiv preprint arXiv:2006.10726*, 2020.

[48] M. Weiler, M. Geiger, M. Welling, W. Boomsma, and T. Cohen. 3d steerable cnns: Learning rotationally equivariant features in volumetric data. *arXiv preprint arXiv:1807.02547*, 2018.

[49] S. J. Wetzel, R. G. Melko, J. Scott, M. Panju, and V. Ganesh. Discovering symmetry invariants and conserved quantities by interpreting siamese neural networks. *Physical Review Research*, 2 (3):033499, 2020.

[50] J. Wu, J. J. Lim, H. Zhang, J. B. Tenenbaum, and W. T. Freeman. Physics 101: Learning physical object properties from unlabeled videos. In *BMVC*, volume 2, page 7, 2016.

[51] T. Yu, C. Finn, A. Xie, S. Dasari, T. Zhang, P. Abbeel, and S. Levine. One-shot imitation from observing humans via domain-adaptive meta-learning. *arXiv preprint arXiv:1802.01557*, 2018.

[52] M. Zhang, H. Marklund, N. Dhawan, A. Gupta, S. Levine, and C. Finn. Adaptive risk minimization: A meta-learning approach for tackling group distribution shift. *arXiv preprint arXiv:2007.02931*, 2020.

[53] Y. D. Zhong, B. Dey, and A. Chakraborty. Symplectic ode-net: Learning hamiltonian dynamics with control. *arXiv preprint arXiv:1909.12077*, 2019.

[54] A. Zhou, T. Knowles, and C. Finn. Meta-learning symmetries by reparameterization. *arXiv preprint arXiv:2007.02933*, 2020.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] In section 5
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] In section 5
   (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? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] In appendix 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)? [No] We plan, however, to open-source our code-based once cleaned.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Specified in the appendix.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] We plot SEM for the video prediction results
   (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] We detail them in each corresponding appendix.
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] We do so in the appendix.
   (b) Did you mention the license of the assets? [Yes] We do so in the appendix.
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Proofs

We utilize the following lemma in the proof of Theorem 1.

Lemma 1. *Fix $v \in \mathbb{R}^d$. Then, for any $\delta > 0$, with probability at least $1-\delta$ over an i.i.d. draw of n examples $((x_i, y_i))_{i=1}^n$, the following holds:*

$$\mathbb{E}_{x,y}[\mathcal{L}(f(x,v),y)] - \frac{1}{n}\sum_{i=1}^{n}\mathcal{L}(f(x_i,v),y_i) \leq C\sqrt{\frac{\ln(1/\delta)}{2n}}.$$

Proof of Lemma 1. By using Hoeffding's inequality, we have that

$$\Pr\left(\mathbb{E}_{x,y}[\mathcal{L}(f(x),y)] - \frac{1}{n}\sum_{i=1}^{n}\mathcal{L}(f(x_i,v),y_i) \geq t\right) \leq \exp\left(-\frac{2nt^2}{C^2}\right),$$

where $t \geq 0$. Solving $\delta = \exp\left(-\frac{2nt^2}{C^2}\right)$ for $t \geq 0$, we have that for any $\delta > 0$, with probability at least $1-\delta$, the following holds:

$$\mathbb{E}_{x,y}[\mathcal{L}(f(x),y)] - \frac{1}{n}\sum_{i=1}^{n}\mathcal{L}(f(x_i,v),y_i) \leq C\sqrt{\frac{\ln(1/\delta)}{2n}}.$$

□

A.1 Proof of Theorem 1

A.1.1 Preparation

In this subsection, we focus on the case of $g_\phi(f_\theta(x)) \neq g_\phi(x)$ as a preparation for the more general case in the next subsection. The (closed) ball of radius r centered at c is denoted by $\mathcal{B}_r[c] = \{v \in \mathbb{R}^d : \|v-c\|_2 \leq r\}$. Fix $r > 0$ and $\mathcal{C}(r,\mathcal{V}) \in \operatorname{argmin}_{\mathcal{C}}\{|\mathcal{C}| : \mathcal{C} \subseteq \mathbb{R}^d, \mathcal{V} \subseteq \cup_{c\in\mathcal{C}}\mathcal{B}_r[c]\}$. Let $\mathcal{N}(r,\mathcal{V}) = |\mathcal{C}(r,\mathcal{V})|$; i.e. the minimum number of balls of radius r needed to cover the a set of vectors $\mathcal{V}$.

The statement of theorem 1 vacuously holds if R is unbounded. Thus, we focus on the case of $R < \infty$ in the rest of the proof. For any $\theta \in \Theta$, there exists $v \in \mathcal{V}$ such that

$$\mathbb{E}_{x,y}[\mathcal{L}(f_\theta(x),y)] - \frac{1}{n}\sum_{i=1}^{n}\mathcal{L}(f_\theta(x_i),y_i) = \mathbb{E}_{x,y}[\mathcal{L}(f(x,v),y)] - \frac{1}{n}\sum_{i=1}^{n}\mathcal{L}(f(x_i,v),y_i) = \psi(v). \tag{4}$$

Moreover, for any $v \in \mathcal{V}$, the following holds: for any $c \in \mathcal{C}(r,\mathcal{V})$,

$$\psi(v) = \psi(c) + (\psi(v) - \psi(c)). \tag{5}$$

For the first term in the right-hand side of (5), by using Lemma 1 with $\delta \to \delta/\mathcal{N}(r,\mathcal{V})$ and taking union bounds, we have that for any $\delta > 0$, with probability at least $1-\delta$, the following holds for all $c \in \mathcal{C}(r,\mathcal{V})$:

$$\psi(c) \leq C\sqrt{\frac{\ln(\mathcal{N}(r,\mathcal{V})/\delta)}{2n}}. \tag{6}$$

By combining equations (5) and (6), we have that for any $\delta > 0$, with probability at least $1-\delta$, the following holds for any $v \in \mathcal{V}$ and all $c \in \mathcal{C}(r,\mathcal{V})$:

$$\psi(v) \leq C\sqrt{\frac{\ln(\mathcal{N}(r,\mathcal{V})/\delta)}{2n}} + (\psi(v) - \psi(c)). \tag{7}$$

This implies that for any $\delta > 0$, with probability at least $1-\delta$, the following holds for any $v \in \mathcal{V}$:

$$\psi(v) \leq C\sqrt{\frac{\ln(\mathcal{N}(r,\mathcal{V})/\delta)}{2n}} + \min_{c\in\mathcal{C}(r,\mathcal{V})}|\psi(v) - \psi(c)|. \tag{8}$$

For the second term in the right-hand side of (8), we have that for any $v \in \mathcal{V}$,

$$\min_{c \in \mathcal{C}(r,\mathcal{V})} |\psi(v) - \psi(c)| \leq \zeta \min_{c \in \mathcal{C}(r,\mathcal{V})} \|v - c\|_2 \leq \zeta r. \tag{9}$$

Thus, by using $r = \zeta^{1/\rho - 1}\sqrt{\frac{1}{n}}$, we have that for any $\delta > 0$, with probability at least $1 - \delta$, the following holds for all $v \in \mathcal{V}$:

$$\psi(v) \leq C\sqrt{\frac{\ln(\mathcal{N}(r,\mathcal{V})/\delta)}{2n}} + \sqrt{\frac{\zeta^{2/\rho}}{n}}. \tag{10}$$

Using equation (4), this implies that for any $\delta > 0$, with probability at least $1 - \delta$, the following holds for all $\theta \in \Theta$:

$$\mathbb{E}_{x,y}[\mathcal{L}(f_\theta(x), y)] - \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(f_\theta(x_i), y_i) \leq C\sqrt{\frac{\ln(\mathcal{N}(r,\mathcal{V})/\delta)}{2n}} + \sqrt{\frac{\zeta^{2/\rho}}{n}}, \tag{11}$$

where $r = \zeta^{1/\rho - 1}\sqrt{\frac{1}{n}}$. Since $\mathcal{N}(r,\mathcal{V}) \leq (2R\sqrt{d}/r)^d = (2R(\zeta^{1-1/\rho})\sqrt{nd})^d$,

$$\begin{aligned}
&\mathbb{E}_{x,y}[\mathcal{L}(f_\theta(x), y)] - \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}(f_\theta(x_i), y_i) \\
&\leq C\sqrt{\frac{\ln(\mathcal{N}(r,\mathcal{V})/\delta)}{2n}} + \sqrt{\frac{\zeta^{2/\rho}}{n}} \\
&= C\sqrt{\frac{\ln(\mathcal{N}(r,\mathcal{V})) + \ln(1/\delta)}{2n}} + \sqrt{\frac{\zeta^{2/\rho}}{n}}, \\
&= C\sqrt{\frac{\ln((2R(\zeta^{1-1/\rho})\sqrt{nd})^d) + \ln(1/\delta)}{2n}} + \sqrt{\frac{\zeta^{2/\rho}}{n}}, \\
&= C\sqrt{\frac{d\ln(\sqrt{d}) + d\ln(2R(\zeta^{1-1/\rho})\sqrt{n}) + \ln(1/\delta)}{2n}} + \sqrt{\frac{\zeta^{2/\rho}}{n}}.
\end{aligned}$$

A.1.2 Putting results together

In this subsection, we now generalize the proof of the previous subsection to the case of $g_\phi(f_\theta(x)) = g_\phi(x)$. The condition of $g_\phi(f_\theta(x)) = g_\phi(x)$ implies that

$$g_\phi^{-1}[\{g_\phi(f_\theta(x))\}] = g_\phi^{-1}[\{g_\phi(x)\}]. \tag{12}$$

Since $g_\phi^{-1}[\{g_\phi(x)\}] = \{x + Az : z \in \mathbb{R}^m\}$ and $f_\theta(x) = x + v_\theta$, the right-hand side of equation (12) can be simplified as

$$g_\phi^{-1}[\{g_\phi(f_\theta(x))\}] = \{f_\theta(x) + Az : z \in \mathbb{R}^m\} = \{x + v_\theta + Az : z \in \mathbb{R}^m\}.$$

Substituting this into equation (12) yields

$$\{x + v_\theta + Az : z \in \mathbb{R}^m\} = g_\phi^{-1}[\{g_\phi(x)\}] = \{x + Az : z \in \mathbb{R}^m\}, \tag{13}$$

where the last equality uses the fact that $g_\phi^{-1}[\{g_\phi(x)\}] = \{x + Az : z \in \mathbb{R}^m\}$. Equation (13) implies that

$$v_\theta \in \mathrm{Col}(A) \subseteq \mathbb{R}^d, \tag{14}$$

where $\mathrm{Col}(A)$ is the column space of the matrix $A \in \mathbb{R}^{d \times m}$. Let $\bar{A} \in \mathbb{R}^{d \times m}$ be a semi-orthogonal matrix such that $\bar{A}^\top \bar{A} = I_m$ (i.e., the identity matrix of size m by m) and $\mathrm{Col}(\bar{A}) = \mathrm{Col}(A)$. Then, equation (14) implies that for any $\theta \in \Theta$, there exists $z \in \mathbb{R}^m$ such that

$$v_\theta = \bar{A}z. \tag{15}$$

We can further refine this statement by using the following observation. Since $\bar{A}^\top \bar{A} = I_m$, we have that

$$R \geq \|v_\theta\|_2 = \|\bar{A}z\|_2 = \|z\|_2, \tag{16}$$

and

$$|\psi(\bar{A}z) - \psi(\bar{A}z')| \leq \zeta \|\bar{A}(z - z')\|_2 = \zeta \|z - z'\|_2. \tag{17}$$

Define $\mathcal{Z} = \{z \in \mathbb{R}^m : \|z\|_2 \leq R\}$. Then, equations (14) and (16) together imply that for any $\theta \in \Theta$, there exists $z \in \mathcal{Z}$ such that

$$v_\theta = \bar{A}z. \tag{18}$$

Whereas the previous subsection defines the ball in the space of $v \in \mathbb{R}^d$, we can now define the ball in the space of $z \in \mathbb{R}^m$ to cover the space of $v \in \mathbb{R}^d$ through equation (18). That is, we re-define $\mathcal{B}_r[c] = \{z \in \mathbb{R}^m : \|z - c\|_2 \leq r\}$. Fix $r > 0$ and $\mathcal{C}(r, \mathcal{Z}) \in \operatorname{argmin}_{\mathcal{C}}\{|\mathcal{C}| : \mathcal{C} \subseteq \mathbb{R}^m, \mathcal{Z} \subseteq \cup_{c \in \mathcal{C}} \mathcal{B}_r[c]\}$. Let $\mathcal{N}(r, \mathcal{Z}) = |\mathcal{C}(r, \mathcal{Z})|$. Using (18), for any $\theta \in \Theta$, there exists $z \in \mathcal{Z}$ such that

$$\begin{aligned}\mathbb{E}_{x,y}[\mathcal{L}(f_\theta(x), y)] - \frac{1}{n}\sum_{i=1}^n \mathcal{L}(f_\theta(x_i), y_i) &= \mathbb{E}_{x,y}[\mathcal{L}(f(x, \bar{A}z), y)] - \frac{1}{n}\sum_{i=1}^n \mathcal{L}(f(x_i, \bar{A}z), y_i) \\ &= \psi(\bar{A}z),\end{aligned} \tag{19}$$

Moreover, for any $z \in \mathcal{Z}$, the following holds: for any $c \in \mathcal{C}(r, \mathcal{Z})$,

$$\psi(\bar{A}z) = \psi(\bar{A}c) + (\psi(\bar{A}z) - \psi(\bar{A}c)). \tag{20}$$

For the first term in the right-hand side of (20), by using Lemma 1 with $\delta \to \delta / \mathcal{N}(r, \mathcal{Z})$ and taking union bounds, we have that for any $\delta > 0$, with probability at least $1 - \delta$, the following holds for all $c \in \mathcal{C}(r, \mathcal{Z})$:

$$\psi(\bar{A}c) \leq C\sqrt{\frac{\ln(\mathcal{N}(r, \mathcal{Z})/\delta)}{2n}}. \tag{21}$$

By combining equations (20) and (21), we have that for any $\delta > 0$, with probability at least $1 - \delta$, the following holds for any $z \in \mathcal{Z}$ and all $c \in \mathcal{C}(r, \mathcal{Z})$:

$$\psi(\bar{A}z) \leq C\sqrt{\frac{\ln(\mathcal{N}(r, \mathcal{Z})/\delta)}{2n}} + (\psi(\bar{A}z) - \psi(\bar{A}c)). \tag{22}$$

This implies that for any $\delta > 0$, with probability at least $1 - \delta$, the following holds for any $z \in \mathcal{Z}$:

$$\psi(\bar{A}z) \leq C\sqrt{\frac{\ln(\mathcal{N}(r, \mathcal{Z})/\delta)}{2n}} + \min_{c \in \mathcal{C}(r, \mathcal{Z})} |\psi(\bar{A}z) - \psi(\bar{A}c)|. \tag{23}$$

For the second term in the right-hand side of (23), by using equation (17), we have that for any $z \in \mathcal{Z}$,

$$\min_{c \in \mathcal{C}(r, \mathcal{Z})} |\psi(\bar{A}z) - \psi(\bar{A}c)| \leq \zeta \min_{c \in \mathcal{C}(r, \mathcal{V})} \|z - c\|_2 \leq \zeta r.$$

If $m = 0$, then since $\mathcal{N}(r, \mathcal{Z}) = 1$ with $r = 0$,

$$\mathbb{E}_{x,y}[\mathcal{L}(f_\theta(x), y)] - \frac{1}{n}\sum_{i=1}^n \mathcal{L}(f_\theta(x_i), y_i) \leq C\sqrt{\frac{\ln(1/\delta)}{2n}}.$$

This proves the desired statement for $m = 0$. Thus, we focus on the case of $m \geq 1$ in the rest of the proof. By using $r = \zeta^{1/\rho - 1}\sqrt{\frac{1}{n}}$, we have that for any $\delta > 0$, with probability at least $1 - \delta$, the following holds for all $z \in \mathcal{Z}$:

$$\psi(\bar{A}z) \leq C\sqrt{\frac{\ln(\mathcal{N}(r, \mathcal{Z})/\delta)}{2n}} + \sqrt{\frac{\zeta^{2/\rho}}{n}}. \tag{24}$$

Using equation (19), this implies that for any $\delta > 0$, with probability at least $1 - \delta$, the following holds for all $\theta \in \Theta$:

$$\mathbb{E}_{x,y}[\mathcal{L}(f_\theta(x), y)] - \frac{1}{n}\sum_{i=1}^n \mathcal{L}(f_\theta(x_i), y_i) \leq C\sqrt{\frac{\ln(\mathcal{N}(r, \mathcal{Z})/\delta)}{2n}} + \sqrt{\frac{\zeta^{2/\rho}}{n}}, \tag{25}$$

where $r = \zeta^{1/\rho-1}\sqrt{\frac{1}{n}}$. Since $\mathcal{N}(r, \mathcal{Z}) \leq (2R\sqrt{m}/r)^m = (2R(\zeta^{1-1/\rho})\sqrt{nm})^m$,

$$
\begin{aligned}
&\mathbb{E}_{x,y}[\mathcal{L}(f_\theta(x), y)] - \frac{1}{n}\sum_{i=1}^{n}\mathcal{L}(f_\theta(x_i), y_i) \\
&\leq C\sqrt{\frac{\ln(\mathcal{N}(r, \mathcal{Z})/\delta)}{2n}} + \sqrt{\frac{\zeta^{2/\rho}}{n}} \\
&= C\sqrt{\frac{\ln(\mathcal{N}(r, \mathcal{Z})) + \ln(1/\delta)}{2n}} + \sqrt{\frac{\zeta^{2/\rho}}{n}}, \\
&= C\sqrt{\frac{\ln((2R(\zeta^{1-1/\rho})\sqrt{nm})^m) + \ln(1/\delta)}{2n}} + \sqrt{\frac{\zeta^{2/\rho}}{n}}, \\
&= C\sqrt{\frac{m\ln(\sqrt{m}) + m\ln(2R(\zeta^{1-1/\rho})\sqrt{n}) + \ln(1/\delta)}{2n}} + \sqrt{\frac{\zeta^{2/\rho}}{n}}.
\end{aligned}
$$

□

A.2 Implications of Theorem 1

We now discuss various cases for the values of m, the dimension of the preimages of g_ϕ:

- (Case of $m = d$) Let us consider an extreme scenario where the function g_ϕ maps all $x \in \mathbb{R}^d$ to one single point. In this scenario, the dimensionality of preimages under g_ϕ is maximized as $m = d$. Accordingly, the bounds with and without enforcing the conservation laws become the same. Indeed, in this scenario, the conservation laws of $g_\phi(f_\theta(x)) = g_\phi(x)$ give us no information, because they always hold for all $x \in \mathbb{R}^d$, even without imposing them.
- (Case of $m = 0$) Let us consider another extreme scenario where the function g_ϕ is invertible. In this scenario, the dimensionality of preimages under g_ϕ is zero as $m = 0$. Thus, imposing the condition of $g_\phi(f_\theta(x)) = g_\phi(x)$ makes the bound in Theorem 1 to be very small. Indeed, in this scenario, the condition of $g_\phi(f_\theta(x)) = g_\phi(x)$ implies that $f_\theta(x) = x$: i.e., x is not moving, and thus it is easy to generalize.
- (Case of $0 < m < d$) From the above two cases, we can see that the benefit of enforcing conservation laws comes from more practical cases in-between these, with $0 < m < d$.

In Theorem 1, the function g_ϕ can differ from the true function g^*_ϕ underlying the system. This is because we analyze a standard generalization gap: i.e., the difference between the expected loss and the training loss. The cost of not using the true g^*_ϕ is captured in the training loss; i.e., the training loss can be large with the function g_ϕ that differs significantly from the true g^*_ϕ. Even in this case, the generalization gap can be small. For example, in the case of $m = 0$, the generalization bound is small, whereas the training loss will be large unless $x_{t+1} = x_t$. Therefore, our analysis gives us the insight on the trade-off between the training loss and the dimensionality of preimages under g_ϕ.

B Related Work

Unsupervised adaptation and meta-learning loss functions. Hand-designed unsupervised losses have been used to adapt to distribution shifts with rotation-prediction [40] or entropy-minimization [47], as well as to encode inductive biases [3]. Unsupervised losses have also been meta-learned for learning to encode human demonstrations [51], few-shot learning exploiting unsupervised information [28, 4], and learning to adapt to group distribution shifts [52]. In contrast, our unsupervised loss only takes the single query we care about, thus imposing no additional assumptions on top of standard prediction problems, and takes the form of a conservation law for predicted sequences.

Encoding of symmetries and physics-based inductive biases in neural networks. We propose to encode symmetries in dynamical systems as meta-learned conserved quantities. This falls under the umbrella of geometric deep learning, which aims to encode symmetries in neural networks; Bronstein et al. [7] provide an excellent review of the field. Most relevant to the types of problems we focus on is roto-translational equivariance [43, 48, 19, 33], applications of GNNs in physical settings [8,

5, 24, 2, 1, 31, 30, 32, 12] and the encoding of time-invariance in RNNs [16, 23, 22, 9, 41]. Recent works have encoded Hamiltonian and Lagrangian mechanics into neural models [21, 27, 10, 18], with gains in data-efficiency in physical and robotics systems, including some modeling controlled or dissipative systems [53, 14]. In contrast to these works, we propose to encode biases via prediction-time fine-tuning following the tailoring framework [3], instead of architectural constraints. This allows us to leverage the generality of loss functions to encode a class of inductive biases extending beyond Lagrangian mechanics, and handle raw video data. Outside mechanics, Suh and Tedrake [39] highlight the difficulty of learning physical inductive biases with deep models from raw data and instead propose to encode mass conservation from pixels with a constrained linear model. Finally, Noether's theorem has been used [20, 42] to theoretically understand the optimization dynamics during learning; unrelated to our goal of discovering inductive biases for sequential prediction.

Discovery of symmetries and conservation laws. There are several previous works which aim to learn conserved quantities in dynamical systems from data. The approach of Schmidt and Lipson [35] focuses on candidate equations that correctly predict dynamic relationships between parts of the system by measuring the agreement between the ratios of the predicted and observed partial (time) derivatives of various components of a system. A set of analytical expressions is maintained and updated as new candidates achieve sufficient accuracy. More recently, Liu and Tegmark [26] discover conservation laws of Hamiltonian systems by performing local Monte Carlo sampling followed by linear dimensionality estimation to predict the number of conserved quantities from an explained ratio diagram. Wetzel et al. [49] learn a Siamese Network that learns to differentiate between trajectories. Contrary to both, Noether Networks do not need segmentations into trajectories with different conserved quantities and can deal with raw pixel inputs and outputs. Other approaches to learning symmetries from data include neuroevolution for learning connectivity patterns [37], regularizers for learning data augmentation [6], and meta-learning equivariance-inducing weight sharing [54]. However, these approaches do not aim to learn conserved quantities in dynamical systems.

Neural networks to discover physical laws. There has been a growing interest in leveraging the functional search power of neural networks for better scientific models. Multiple works train a neural network and then analyze it to get information, such as inferring missing objects via back-propagation descent [2], inspecting its gradients to guide a program synthesis approach [44], learning a blue-print with a GNN [11], or leveraging overparametrized-but-regularized auto-encoders [45]. Others, such as DreamCoder [15], take the explicit approach with a neural guided synthesis over the space of formulas. Unlike these works, our method can readily be applied both to symbolic conservation laws and to neural network conservation losses applied to raw videos.

C Experimental details for scientific data

We use three different experiments from [21]: the ideal pendulum, the ideal spring, and the real pendulum, under an Apache license. The first two come from simulated ODEs without noise, the latter comes from a real data from Schmidt and Lipson [35]. It is worth noting that other datasets from Greydanus et al. [21] (such as a planetary system) could not be included because the DSL required too much depth to reach the conserved energy quantity. A better search, such as using evolution, or a better DSL (such as those derived from many scientific formulas in DreamCoder [15]) could remedy this. Finally, note that an approach that searched over the same DSL encoding the loss as a generic $f(x_0, x_t)$ loss, not as a conservation $|g(x_0) - g(x_t)|$ would require more than twice the depth, thus not being able to cover the evaluated datasets.

For the ideal pendulum, input $x = (p, q) \in \mathbb{R}^2$ contains the angle q and momentum p of the pendulum. The formula for the energy is $\mathcal{H} = 2mgl(1 - \cos q) + \frac{l^2 p^2}{2m}$. Greydanus et al. [21] set $m = \frac{1}{2}, l = 1, g = 3$, resulting in $\mathcal{H} = 3(1 - \cos q) + p^2$. Notice that a simpler conserved quantity is $p^2 - 3 \cdot \cos q$.

For the spring, input $x = (p, q) \in \mathbb{R}^2$ contains the displacement q and momentum p, and the system's energy is given by $\mathcal{H} = \frac{1}{2}kq^2 + \frac{p^2}{2m}$. Greydanus et al. [21] set $k = q = m = 1$, resulting in $\mathcal{H} = \frac{1}{2}(q^2 + p^2)$, where units are omitted. Thus, $q^2 + 1 \cdot p^2$ is a conserved quantity, with 1 having the appropriate units.

Since formulas in the DSL have physical meaning, each sub-formula carries its own associated physical units. When applying an operation, there is a check for validity (we can only input degrees to a sinusoid, we can only add quantities of the same units, etc.). This allows us to significantly prune the exponential space, as done in AI-Feynman [44]. The vocabulary of the DSL is the following: `Input(i)`: returns the i-th input, `Operation`: one of $\{+, -, \cdot, /, \sin, \cos, x^2\}$, `Parameter(u)`: trainable scalar with units $[u]$.

The number of formulas increases exponentially with the number of terms involved, leading to vast search spaces. To save computation, we perform a pipelined approach that first generates all valid formulas up to depth 7. This removes formulas whose operations take in incompatible physical units and prunes equivalences of commutativity for the appropriate operations. We obtain 41,460 and 72,372 formulas for the pendulum and spring, respectively. Note that we do *not* force the conserved quantity to have units of energy since in general not all conserved quantities are energies.

The resulting formulas are purely symbolic, with some trainable parameters still to be defined. We check that they are approximately conserved by optimizing their parameters via gradient descent to have low variance within sequences of true data. We then do the same for random sequences; if conservation is two orders of magnitude smaller for the true data, we accept it as an approximate conservation. This measure is similar to others used before for conserved quantities [26], but it has problems when the data is noisy, the minimisation is sub-optimal, or there are numerical issues. As a result, we obtain 210 and 219 approximately conserved quantities for the pendulum and spring, respectively.

Finally, we evaluate candidates expression for their usefulness as Noether losses. For each candidate we try one inner step with a range of inner learning rates 10^k for $k \in \{-3, -2.5, -2, \dots, 1\}$. To save compute, we first obtain the top 20 candidate equations by the train performance when tailoring the vanilla MLP with each loss. For those 20 candidates, we try them as meta-tailoring losses: starting from the pre-trained vanilla MLP, we fine-tune it for 100 epochs using meta-tailoring with each candidate Noether loss. We then evaluate the MSE using the fine-tuned model for long-term predictions, keep the expression with the best loss in the training data, and report the long-term prediction on the test data.

It is worth noting that we use a pipeline approach: first discovering approximately conserved quantities, then pruning a subset of 20 that are most useful for tailoring and then finally pick the best loss when used with meta-tailoring. This pipeline, along with the concrete numbers of 20 candidates and 2 orders of magnitude comparing variance in real data vs. random data, were heuristically chosen to speed up the search, without trying other hyper-parameters. For the real pendulum we found that two losses that were among the top 20 for tailoring diverged when used for meta-tailoring. We conjecture this is the case because of the noise in the real data

For the ideal pendulum, in addition to the true formula, our method also finds equivalent losses that the initial pruning did not detect, such using a sum with a negative parameter: $p^2 + (-2.99) \cdot \cos(q)$ instead of $p^2 - 2.99 \cdot \cos(q)$.

For the real pendulum, it is worth noting that the energy candidate $p^2 - 2.4\cos(q)$ was proposed by humans to fit the data, since the data is experimental. It could thus be slightly sub-optimal, which could explain why Noether Networks improves its predictions. We also consider the possibility of it being due to fluctuations in noisy data and it being a small dataset.

Experiments were performed on an NVIDIA Tesla V-100 GPU and 10 CPU cores, taking around 4 hours to run.

D Experimental Details for Physics 101

We use a subset of the Physics 101 dataset [50] with videos from the ramp scenario, where various objects are let go by a human hand at the top of a ramp. We could not find its associated license. There are two ramp settings: 10 degrees and 20 degrees. To ensure the prediction problem is interesting and nontrivial, we only take examples with the 20 degree ramp since the object does not slide down the ramp in many of the 10 degree examples. Additionally, we only take sequences with length at least 2 seconds, and use a frame rate of 15 frames per second when extracting frames to pass to the models. We use videos recorded with the `Kinect_RGB_1` sensor. Video frames are center-cropped at full height (1080 pixels) and downsample to obtain 128×128 images, and perform random horizontal

flipping for sequences. This preprocessing results in 389 possible sequences, and we take a random 80/20 train/test split. In the prediction task, we condition the model on two frames, and the model must predict the subsequent 20 frames. To ensure that we extract segments of videos where the object is moving, we take the middle 22 frames of each video clip. We use a mini-batch size of 2 for these experiments.

Our baseline model is the publicly-available implementation of SVG-LP [13], with the batch normalization layers replaced with layer normalization layers. This change is made because of the small mini-batch size used in our experiments. As in the original implementation, the encoder and decoder are based on VGG16 [36], the latent frame predictor is a 2-layer LSTM with hidden size of 128, and both the prior and posterior are single-layer LSTMs that produce latent variables $\mathbf{z}_t$ of dimension 64. The learning rate used during the training of the baseline 0.0001. Teacher forcing is used during the training of both the SVG baseline — as done by Denton and Fergus [13] — as well as the Noether Network's embedding.

The meta-learned inner loss of the Noether Network is defined in Equation 2, with the learned embedding g_ϕ parameterized as a 2-layer convolutional network where each layer consists of a 5×5 convolution with 32 filters in the first layer and 64 filters in the second layer, a ReLU non-linearity, and 2×2 max pooling. There is a final linear layer which projects onto a 64-dimensional embedding space, in which the MSE is computed for the inner loss.

The CNGRAD algorithm of Alet et al. [3] is used to perform meta-tailoring. In the Noether Network, conditional normalization (CN) layers are inserted after each layer normalization layer in both the encoder and the decoder. The CN parameters are initialized to the identity transformation, and are adapted with a single inner step at prediction-time using an inner learning rate of 0.0001. The embedding is trained in the outer loop with an outer learning rate of 0.0001, along with the rest of the model parameters, excluding CN parameters. We report results obtained with meta-tailoring, where all weights are randomly initialized. In the reported results, all models are trained for 400 epochs, at which point training has converged.

We ran all experiments with the Physics 101 dataset on an NVIDIA Tesla V100 GPU and 20 CPU cores. Training the baseline until convergence took approximately 13 hours, and training the Noether Network took approximately 2 days, 23 hours.