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]
MIT[1], Stanford University[2], National University of Singapore[3]
{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 [18], permutation invariance in Graph Neural Networks [26] for drug design [31], or roto-translational equivariance in SE3-transformers [14] for protein structure prediction [17]. However, for data distributions of interest, there may be exploitable symmetries that are either unknown or difficult to describe with code. Progress has been made in automatically discovering symmetries for finite groups [44], but meta-learning and exploiting general continuous symmetries has presented a major challenge. In part, this is because symmetries describe the effect of counterfactuals about perturbations to the data, which are not directly observable.

In this work, we propose to exploit symmetries in sequential prediction problems indirectly. We take inspiration from Noether's theorem [22], 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.

For example, consider a system of planets interacting via gravity: this system is translation invariant in all three cardinal directions (i.e. translating the entire system in the x,y, or z axis conserves the laws of motion). Noether's theorem asserts there must be a conserved quantity for each of these symmetries; in this case, linear momentum. Similarly, the system has a time-invariance (i.e. the laws of motion are the same today as they will be tomorrow). In this case, the corresponding conserved quantity is the total energy of the system.

[*]Equal contribution.

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

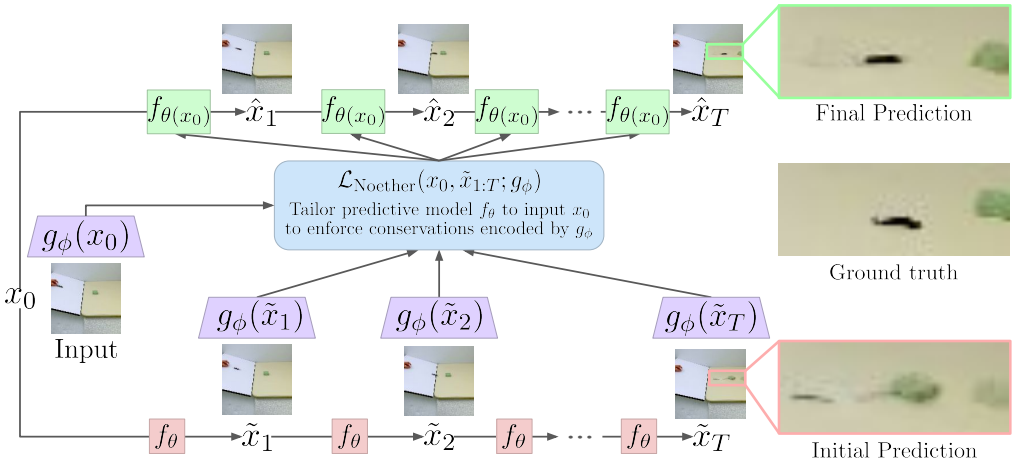


Figure 1: Noether Networks enforce conservation laws, meta-learned by g_ϕ, in sequential predictions made by f_θ, which is tailored to the input x_0 to produce final predictions $\hat{x}_{1:T}$. Imposing these meta-learned inductive biases improves video prediction quality with objects sliding down a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

Inspired by this equivalence, we propose that approximate conservation laws are a powerful paradigm for meta-learning useful inductive biases in sequential prediction problems. Whereas symmetries are difficult to discover because they are global properties linked to counterfactuals about unobserved perturbations of the data, conserved quantities can be directly observed in 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 [2], which proposes to encode inductive biases by fine-tuning neural networks with hand-designed unsupervised losses inside the prediction function. Whereas traditional auxiliary losses are added to the task loss during training, tailoring losses are optimized inside the prediction function both during training and testing, customizing the model to each individual query. In doing so, we ensure there is no generalization gap for the conservation loss. We propose to meta-learn the self-supervised loss function and parameterize it in the form of a conservation loss; i.e. $\mathcal{L}(x_0, \tilde{x}_{1:T}; g_\phi) = \sum_{t=1}^{T} |g_\phi(x_0) - g_\phi(\tilde{x}_t)|^2$. This conservation form encodes a meta-inductive bias (inductive bias over inductive biases) which narrows the search space exponentially (in T) and simplifies the parameterization. Figure 1 demonstrates the Noether Network pipeline for a video prediction task.

The main contribution of this paper is the Noether Network, an architecture class and algorithm that can automatically learn its own inductive biases in the form of meta-learned conservation loss functions. We theoretically characterize the advantages of such conservation laws as effective regularizers that constrain the learning to lower-dimensional output manifolds, lowering the generalization gap. Empirically, we first find that, when the meta-learned conservation loss takes the form of a synthesized program, Noether Networks recover known conservation laws from raw physical data. Second, we find that Noether Networks can learn conservation losses on raw videos, modestly improving the generalization of a video prediction model, especially on longer-term predictions.

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. We now discuss various cases for the values of m:

- (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.

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 framework [2]. 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 minimizes 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. In contrast to auxiliary losses, which would optimise the conservation loss only for the training points, tailoring allows us to ensure conservation at test time. Since we aim to build in the conservation in the architecture, we want to ensure it is also satisfied for unseen test samples. Another advantage of tailoring losses is that they are easier to meta-learn. Auxiliary losses are pooled over all examples and training epochs and their effect is only known at validation/test time. We would need to use implicit gradients [20, 24] to know their eventual effect on the final weights at the end of training. With tailoring, we can directly measure the effect of the meta-learned update on the same sample.

A limitation of tailoring framework 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. Our approach can be seen as a generalization of tailoring where the unsupervised loss is meta-learned and takes the form of a conservation loss.

Noether Networks: discovering *useful* conservation laws. We build on the paradigm of conservation laws; i.e. quantities that are conserved over time. This has the following two challenges:

1. Real data often has noise, breaking any exact conservation law. Moreover, many conservation laws are only approximately satisfied given only partial information of the system. For example, conservation of energy is not fully satisfied in real dissipative systems and estimating conservation of mass from pixels is bound to be inexact in the case of occlusions.
2. Optimizing for conservation alone can lead to trivial quantities, such as predicting a constant value $g_\phi(x) = C$ independent of x.

We search for *useful* conservation laws, whose (approximate) enforcing brings us closer to the true data manifold for the current sequence. Note that a useful conserved quantity doesn't need to be exactly conserved in the training data to improve prediction performance. We only need its conservation to be informative of the data manifold. This allows us to discover conservation laws even in imperfect, noisy environments. Conversely, tailoring with a trivial conserved quantity cannot improve the final predictions (see App. B for a detailed explanation). Finally, viewing conserved quantities as useful inductive biases aligns well with their use, since we often care about biases only insofar as they help improve task performance.

The search for such useful conserved quantities can take many forms. In this work, we focus on two: first, in the beginning of section 4.1, we use a combination of program synthesis and gradient descent to generate a large, but finite, number of candidate parametric formulas for physical system prediction. We then try meta-tailoring with each formula on the training set and pick the conservation formula with the best loss. Formulas can be a useful, interpretable description in scientific domains given the appropriate state-space. However, for general descriptions from raw data, we would like to describe the loss with a neural network, as we describe in the next section.

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

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

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 [12], as well as works on meta-learning loss functions for few-shot learning [3] and group distribution shift [41]. Algorithm 1 provides pseudo-code.

For deep learning frameworks that allow per-example weights, such as JAX [5], the loop over sequences in Alg. 1 can be efficiently parallelized. To parallelize it for other frameworks we use the CNGRAD algorithm [2], which adapts only the Conditional Normalization (CN) layers in the inner loop. Similar to BN layers, CN only performs element-wise affine transformations: $y = x \odot \gamma + \beta$, which can be efficiently parallelized in most deep learning frameworks even with per-example γ, β.

Finally, even though we use the two-layer optimization typical of meta-learning, we are still in the classical single-task single-distribution setting. Noether Networks learn to impose their own inductive biases via a learned loss and to leverage it via an adaptation of its parameters θ. This is useful as we often do not have access to meta-partitions that distinguish data between distributions or tasks.

Algorithm 1 Prediction procedure 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($x_0; \theta, \phi$)
  $\tilde{x}_0, \hat{x}_0 \leftarrow x_0, x_0$
  $\tilde{x}_t \leftarrow f_\theta(\tilde{x}_{t-1})\ \forall t \in \{1, \ldots, T\}$ ▷ 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
  $\hat{x}_t \leftarrow f_{\theta(x_0;\phi)}(\hat{x}_{t-1})\ \forall t \in \{1, \ldots, T\}$ ▷ 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 $x^{(0)}_{0:T}, \ldots, x^{(N)}_{0:T} \sim \mathcal{D}_{\text{train}}$
    for $0 \leq n \leq N$ **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

4 Experiments

Our experiments are designed to answer the following questions:

1. Can Noether Networks recover known conservation laws in scientific data?
2. Are Noether Networks useful for settings with controlled dynamics?
3. Can Noether Networks parameterize useful conserved quantities from raw pixel information?
4. How does the degree of conservation affect performance?

4.1 Experimental domains

To answer the above questions, we consider a number of different experimental domains, which we overview in this subsection, before moving on to the evaluation and results.

Spring and pendulum from state coordinates. Greydanus et al. [16] propose the setting of an ideal spring and ideal pendulum, which will allow us to understand the behavior of Noether Networks for scientific data where we know a useful conserved quantity: the energy. They also provide data from a real pendulum from [27]. In contrast to the ideal setups, here the data is noisy and the Hamiltonian is only approximately conserved, as the system is dissipative. For the two pendulum, input $x = (p, q) \in \mathbb{R}^2$ contains its angle q and momentum p. Given the parameters in [16] the energy is $\mathcal{H} = 3(1 - \cos q) + p^2$, with $p^2 - 3 \cdot \cos q$ being a simpler equivalent. For the spring, input $x = (p, q) \in \mathbb{R}^2$ contains the displacement q and momentum p and the energy is $\mathcal{H} = \frac{1}{2}(q^2 + p^2)$. Thus, $q^2 + 1.0 \cdot p^2$ is a conserved quantity, where the coefficient 1.0 has appropriate units.

We build on their vanilla MLP baseline and discover conservation laws that, when used for meta-tailoring, improve predictions. Since the baseline predicts $\frac{d}{dt}(x_t)$ rather than x_{t+1}, we apply the loss to a finite-difference approximation, i.e. $\mathcal{L}_{\text{Noether}}\left(x_t, x_t + \frac{dx}{dt}\Delta_t\right) = \mathcal{L}_{\text{Noether}}\left(x_t, x_t + f_\theta(x_t)\Delta_t\right)$.

Domain specific language for scientific formulas. To search over Hamiltonians, we program a simple domain specific language (DSL) for physical formulas. Since formulas in the DSL have physical meaning, each sub-formula carries its own associated physical units and is checked for validity. This allows us to significantly prune the exponential space, as done in AI-Feynman [35]. 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 $[u]$.

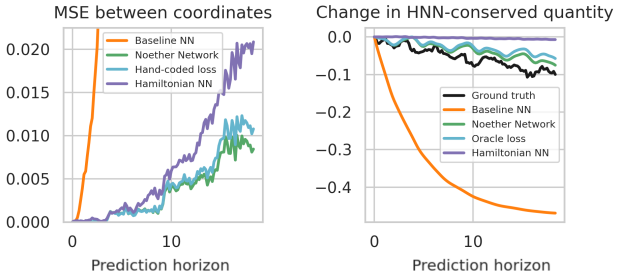
Table 1: Recovering $\mathcal{H}$ for the ideal pendulum.

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.



Pixel pendulum with controls. In settings such as robotics we may be interested in *action-conditioned* video prediction [23]: predicting future video frames given the history of frames and an agent's actions. We recorded videos of episodes in OpenAI Gym's pendulum swing up environment, using a scripted policy to control the pendulum. Each model receives four history frames and a sequence of 26 policy actions starting from the current timestep, and must predict 26 future frames. As this is a visually simple environment and we wish to test generalization, the training dataset has 5 episodes of 200 frames each. We hold out 195 episodes for testing.

Video prediction with real-world data. To characterize the benefits provided by Noether Networks in a real-world video prediction setting, we compare a Noether Network version of the stochastic video generation (SVG) model [9] — a VAE-based method — to a standard SVG baseline on the ramp scenario of the Physics 101 dataset [39]. 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. To increase difficulty, we restrict our data to the 20 degree incline and to sequences of at least two seconds.

4.2 Evaluation

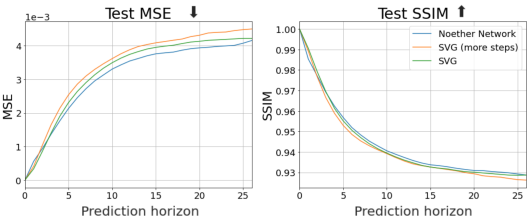
Can Noether Networks recover known conservation laws in scientific data? We first generate all valid formulas up to depth 7, removing operations between incompatible physical units and formulas equivalent up to commutativity. We obtain 41,460 and 72,372 candidate conserved formulas for the pendulum and spring, respectively, which need *not* have units of energy.

The resulting formulas are purely symbolic, with some trainable parameters still to be defined. We thus optimize 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 [19], but is not sufficient on its own when the data is noisy, the minimisation is sub-optimal, or there are numerical issues. As a result, we obtain 210 and 219 candidates for approximately conserved quantities for the pendulum and spring, respectively.

Finally, for each potentially conserved quantity, we try it as meta-tailoring loss: starting from the pre-trained vanilla MLP, we fine-tune it for 100 epochs using meta-tailoring, with one inner step and a range of inner learning rates 10^k for $k \in \{-3, -2.5, \ldots, 1\}$. We then evaluate the MSE of the fine-tuned model for long-term predictions, keeping the expression with the best loss in the training data.We observe that this process correctly singles out equations of the same form as the true Hamiltonian $\mathcal{H}$, with almost the exact parameters. Using these as losses reaches equivalent performance with that of the oracle, which uses the true formula (see Tables 1 and 2).

Finally, we run the same process for a real pendulum [27], where energy is not conserved. We use largely the same pipeline as for the ideal pendulum, the differences are explained in Appendix C. Noether Networks discover $p^2 - 2.39\cos(q)$, close to the (potentially sub-optimal) $\mathcal{H} = p^2 -$

Figure 3: In controlled pendulum environment, the Noether Network has lower mean squared error (left) and slightly better structural similarity (right). Metrics are computed and plotted by prediction timestep, where timestep 0 is a given history frame. Note that simply training SVG for more steps does not increase performance.



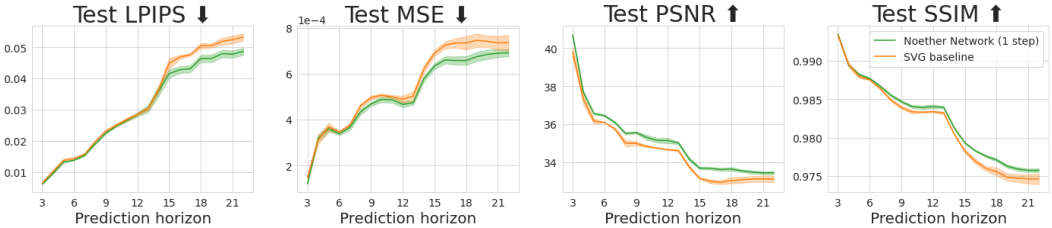


Figure 4: The Noether Network outperforms the baseline by a small margin in all four metrics in real-world video prediction, showing they can meta-learn useful conserved quantities from raw pixel data.

$2.4\cos(q)$ described in [16]. This Noether Network improves the baseline 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 conservation of energy. Results can be seen in figure 2.

Are Noether Networks useful in settings with controlled dynamics? For the controlled pendulum experiments, we begin with an SVG model but modify it to (1) remove the prior sampling component since the environment is deterministic given actions and (2) concatenate each action to the corresponding frame encoder output and feed the result to the LSTM. After training the SVG model for 50 epochs and fixing its weights, we run Algorithm 1 for 20 epochs to learn an embedding network for conserving useful quantities. We found that Noether Networks perform better in this setting by directly adapting the latent LSTM activations against the conservation loss, rather than adapting any network parameters. Since Noether Networks take additional training steps starting from a pre-trained SVG model, we also continued training the SVG model for 20 further epochs, which we call "SVG (more steps)." Figure 3 shows the results of each method on held out sequences. Noether Networks improve the overall mean squared error (MSE) and structural similarity (SSIM), more so than simply training SVG further.

Can Noether Networks parameterize useful conserved quantities from raw pixel information? In our experiments, we train on only 311 training sequences, which makes generalization 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 base model from scratch for 400 epochs. As seen in figure 4, taking a single inner tailoring step improves performance slightly over the baseline SVG model with respect to learned perceptual image patch similarity (LPIPS) [42], mean squared error (MSE), peak signal-to-noise ratio (PSNR), and structural similarity (SSIM). As the prediction horizon increases, the Noether Network performs comparatively better than the baseline, likely because imposing conservations encourages the model to retain aspects of the ground-truth input frames. These results provide some evidence that Noether Networks are capable of learning useful inductive biases from raw video data.

To investigate whether the Noether embeddings learn relevant features, we use Gradient-weighted Class Activation Mapping (Grad-CAM) [28] to compute localization maps for sequences in the test set which show frame regions that are "important" to each dimension of the embedding. Since the Noether embedding for the Physics 101 experiments has 64 dimensions, we perform principal component analysis (PCA) before Grad-CAM to reduce the dimensionality of the embedding, and to sort dimensions by the percentage of variance they explain in the test set frames. Interestingly, we find that the first PCA dimension captures 83.6% of the variance, and the first four dimensions capture 99.9% of the variance. Figure 5 shows Grad-CAM localization maps for two example

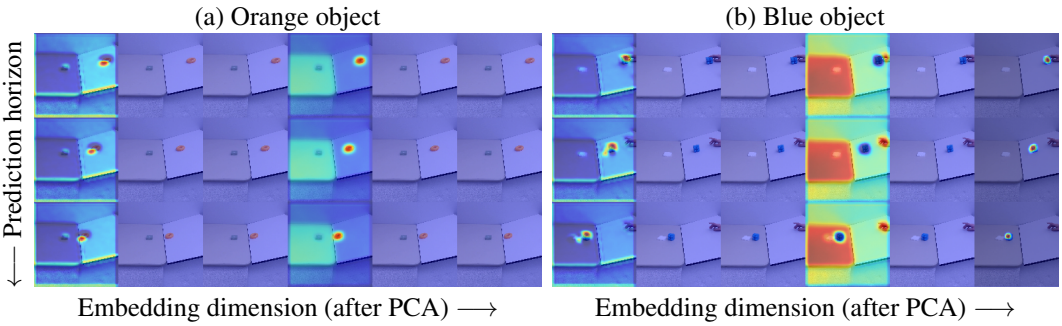


Figure 5: Grad-CAM localization maps show that the Noether embeddings attend to relevant frame regions for two example sequences. Heatmaps for the first six dimensions of the embedding (in the PCA basis, sorted by decreasing amount of variance explained) are shown for three time steps.

test-set sequences, (a) depicts an orange torus and (b) shows a blue brick, where warmer colors (red) indicate high importance and cooler colors (blue) indicate low importance. The first dimension, which explains the vast majority of the variance, primarily focuses on the sliding object in both examples. It also attends to the object on the table, and to the edge of the ramp. The attention to the objects suggests that the Noether Network learns to conserve quantities related to the objects' pixels and their motion. We hypothesize that the attention to the table edge encodes information about the orientation of the sequence, since half of the training sequences are randomly flipped horizontally. While several of the dimensions focus nearly uniformly on the entire frame (thus conveying little information), the fourth dimension focuses on the orange sliding object in (a), and the human hand and table in (b). In all the examples in the test set, whenever the human hand is in the frame, it is attended to by this dimension. We hypothesize that it learns to conserve the hand in sequences where it is present, and possibly picks up on similarly-colored pixels as well. Grad-CAM localization maps are shown for additional example sequences in Appendix F.

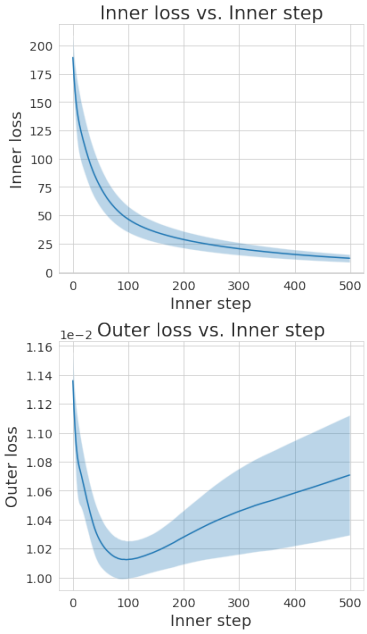


Figure 6: Despite only being trained with one inner step, as we bring the inner Noether loss to zero (top), the outer loss improves for a long period of adaptation (bottom).

How does the degree of conservation affect performance? All of the results presented in this work were achieved by Noether Networks trained and evaluated with a single inner step. To characterize how the degree to which the conservations are imposed affects video prediction performance, we optimize the inner (Noether) loss for many inner steps and measure the outer (task) loss, as shown in figure 6. Here, the inner loss is optimized by Adam for 500 steps, as opposed to the single step of SGD during training (both settings use a learning rate of 10^{-4}). During the optimization, the outer loss continues to improve for roughly 100 inner steps, which suggests that the approximate conservations learned by the Noether Network generalize to more exact conservation settings.

5 Related Work

Unsupervised adaptation and meta-learning loss functions. Hand-designed unsupervised losses have been used to adapt to distribution shifts with rotation-prediction [33] or entropy-minimization [37], as well as to encode inductive biases [2]. Unsupervised losses have also been meta-learned for learning to encode human demonstrations [40], few-shot learning to exploit test-data as unsupervised information [3], and learning to adapt to group distribution shifts [41]. 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.

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 [27] 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 [19] 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. [38] 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 [30], regularizers for learning data augmentation [4], and meta-learning equivariance-inducing weight sharing [44]. 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 [1], inspecting its gradients to guide a program synthesis approach [35], learning a blue-print with a GNN [8], or leveraging overparametrized-but-regularized auto-encoders [36]. Others, such as DreamCoder [11], 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.

Encoding physics-based inductive biases as architectural constraints. Recent works have encoded Hamiltonian and Lagrangian mechanics into neural models [16, 21, 7, 13], with gains in data-efficiency in physical and robotics systems, including some modeling controlled or dissipative systems [43, 10]. In contrast to these works, we propose to encode biases via prediction-time fine-tuning 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 [32] highlight the difficulty of learning physical inductive biases with deep models from raw data and instead propose to encode mass conservation in a video prediction model with a constrained linear model. More generally, Bronstein et al. [6] provide an excellent review of the use of symmetries for deep learning. Finally, Noether's theorem has been used [15, 34] to theoretically understand the optimization dynamics during learning; unrelated to our goal of discovering inductive biases for sequential prediction.

6 Discussion

Limitations. A disadvantage of our proposed approach over architectural constraints (in the cases when the latter can represent the same bias) is computation time. Noether Networks optimize the conservation loss inside the prediction function, which results in an overhead. This can be reduced by only fine-tuning the higher-level layers, avoiding back-propagation back to the input. Another approach to reduce training time could be to warm-start the Noether embeddings with pre-trained contrastive embeddings, which have an aligned goal and have shown great results for video and audio classification [25]. It is also sometimes the case that, by encoding the inductive bias during training the trained but un-tailored Noether Network performs better than a comparable architecture without the bias during training. It is also worth noting that the theoretical guarantees apply to fully enforced conservation constraints, while in practice we only fine-tune our model to better satisfy them. However, we believe the theoretical result still conveys the potential benefits of Noether networks.

Broader impact. We propose a new framework for meta-learning inductive biases in sequential settings, including from raw data. On the positive side, our framework can be used to discover conserved quantities in scientific applications, even when these are not fully conserved. By improving video prediction, we are also empowering predictive models for entities such as robots and self-driving cars, helping elder care and mobility. However, at the same time, it could be used for better predicting the movement and tracking of people from CCTV cameras, affecting privacy. It could also be used to improve the quality of deepfake video generation, which can be used for malicious purposes.

Overview and future work. We propose Noether Networks: a new framework for meta-learning inductive biases through 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, 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.

[2] 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.

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

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

[5] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

[6] 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.

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

[8] 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.

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

[10] 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.

[11] 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.

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

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

[14] 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.

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

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

[17] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

[18] 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.

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

[20] J. Lorraine, P. Vicol, and D. Duvenaud. Optimizing millions of hyperparameters by implicit differentiation. In *International Conference on Artificial Intelligence and Statistics*, pages 1540–1552. PMLR, 2020.

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

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

[23] J. Oh, X. Guo, H. Lee, R. Lewis, and S. Singh. Action-conditional video prediction using deep networks in atari games. *arXiv preprint arXiv:1507.08750*, 2015.

[24] A. Raghu, M. Raghu, S. Kornblith, D. Duvenaud, and G. Hinton. Teaching with commentaries. *arXiv preprint arXiv:2011.03037*, 2020.

[25] A. Recasens, P. Luc, J.-B. Alayrac, L. Wang, F. Strub, C. Tallec, M. Malinowski, V. Patraucean, F. Altché, M. Valko, et al. Broaden your views for self-supervised video learning. *arXiv preprint arXiv:2103.16559*, 2021.

[26] 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.

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

[28] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

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

[30] 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.

[31] 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.

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

[33] 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.

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

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

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

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

[38] 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.

[39] 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.

[40] 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.

[41] 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.

[42] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

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

[44] 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 6
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] In section 6
   (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)? [Yes] Code can be found at https://lis.csail.mit.edu/noether.
   (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)? [No] Some plots are missing error bars due to computational constraints. For the Physics-101 dataset we ran each method with 5 seeds and report the SEM.
   (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 $\mathcal{C}(r,\mathcal{V}) \in \mathrm{argmin}_{\mathcal{C}}\{|\mathcal{C}| : \mathcal{C} \subseteq \mathbb{R}^d, \mathcal{V} \subseteq \cup_{c\in\mathcal{C}}\mathcal{B}_r[c]\}$. 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 $\mathcal{N}(r, \mathcal{Z}) \leq (2R\sqrt{m}/r)^m = (2R(\zeta^{1-1/\rho})\sqrt{nm})^m$,

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

□

B Why Noether Networks avoid learning trivial conserved quantities

Neural networks have large capacity. Therefore, we have to be careful that the Noether embedding is not learning a quantity that is trivially conserved but is unrelated to the task, such as predicting a constant vector $g_\phi(x) = \vec{C}$. In our setup both the prediction network and the meta-learned conservation embedding are trained end-to-end to minimize the task loss after the conservation update. However, for the meta-learned embedding this is equivalent to encouraging a big decrease in supervised loss after the update than before the update. This can be seen by looking at the gradient of the difference between the task loss before and after the update:

$$\nabla_\phi \left(\mathcal{L}_{\text{task}}(x_{1:T}, f_{\theta(x_0;\phi)}) - \mathcal{L}_{\text{task}}(x_{1:T}, f_\theta)\right)$$

Here $\theta(x_0; \phi)$ denotes the model parameters after the inner lo(update which conserves the embedding g_ϕ for input x_0. Notice that the second loss does not depend on the embedding (since θ has not yet been tailored with g_ϕ). Thus, the gradient of the overall expression is equivalent to:

$$\nabla_\phi \left(\mathcal{L}_{\text{task}}(x_{1:T}, f_{\theta_{x_0}(\phi)})\right)$$

which is the one we optimize. A trivially conserved quantity (like $g_\phi(\cdot) = 0$ independent of θ, x) will not produce any improvement on the supervised loss after its conservation is encouraged. Therefore, the embedding will seek to be more useful, by making the tailored weights $\theta(x_0; \phi)$ perform better than the untailored ones θ.

C Experimental details for scientific data

We use three different experiments from [16]: 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 [27]. It is worth noting that other datasets from Greydanus et al. [16] (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 [11]) 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.

Most experimental details and hyperparameters are already detailed in the main text. It is worth noting that the 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.

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

Extra details on state space and energies For the pendulum, input $x = (p, q) \in \mathbb{R}^2$ contains its angle q and momentum p. The formula for the energy is $\mathcal{H} = 2mgl(1 - \cos q) + \frac{l^2 p^2}{2m}$. Greydanus et al. [16] 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. [16] 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, where coefficient 1 has appropriate units.

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.

D Experimental Details for pendulum with controls

We generated the action-conditioned video prediction data in OpenAI Gym's "Pendulum-v0" environment. We want to make each episode different and not easily predictable, but also not too erratic. Hence we generate actions for each episode by:

$$a_t = 2\sin(\omega t + \alpha)$$

where ω, α are randomly sampled for each episode, with $\omega \sim \mathrm{U}[0.05, 0.15]$ and $\alpha \sim \mathrm{U}[0, 2\pi]$. We recorded 200 episodes of 200 timesteps each and split them into 5 train and 195 test videos.

Our action conditioned video prediction architecture is based off of the publicly available implementation[2] of SVG, with prior sampling removed and actions concatenated to the output of the frame encoder. The frame encoder and decoder are based on the DCGAN generator and discriminator, and the latent frame predictor is a 2-layer LSTM with hidden size 256. We resize the input frames to 64×64 before inputting to the frame encoder. Throughout all experiments we use a batch size of 32 and optimize using Adam with a learning rate of 0.001.

During evaluation, each method receives a length-4 frame history and 26 future actions and must predict the next 26 frames. However, during training each method only predicts the next 10 frames and then is updated to minimize mean square error; this speeds up each step and reduces memory cost. For the Noether Network there is an inner loop where we minimize the learned conservation loss: in these experiments, we found it better to update the LSTM's initial outputs during the inner loop, rather than any network parameters. We then pass the updated LSTM outputs into the frame decoder to generate a video prediction. The inner loop optimization is done with gradient descent using a learning rate of 0.5, for 1 step. As described above the conservation loss involves conserving a learned embedding over time. Our learned embedding in this case is a 3-layer MLP with 512 units at each hidden layer and outputs an embedding with dimension 100. The embedding receives as input the LSTM output for the past two time steps and the most recent action, all concatenated together.

We ran all the action conditioned experiments on a single machine with an NVIDIA RTX 2080 Ti GPU. Due to the limited data available, all training was relatively fast: the initial 50 epochs of SVG training took ~ 40 minutes, while the Noether Network tuning process took ~ 17 minutes.

E Experimental Details for Physics 101

We use a subset of the Physics 101 dataset [39] 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

[2] https://github.com/edenton/svg

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 based on the publicly-available implementation of SVG-LP [9]. As in the original implementation, the encoder and decoder are based on VGG16 [29], 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. We replace all batch normalization layers with layer normalization layers in both the baseline and the Noether Network because of the small mini-batch size used in our experiments. The learning rate used during the training of the baseline 0.0001. Teacher forcing is used during the training of the SVG baseline and the meta-training of the Noether Network, as done by Denton and Fergus [9].

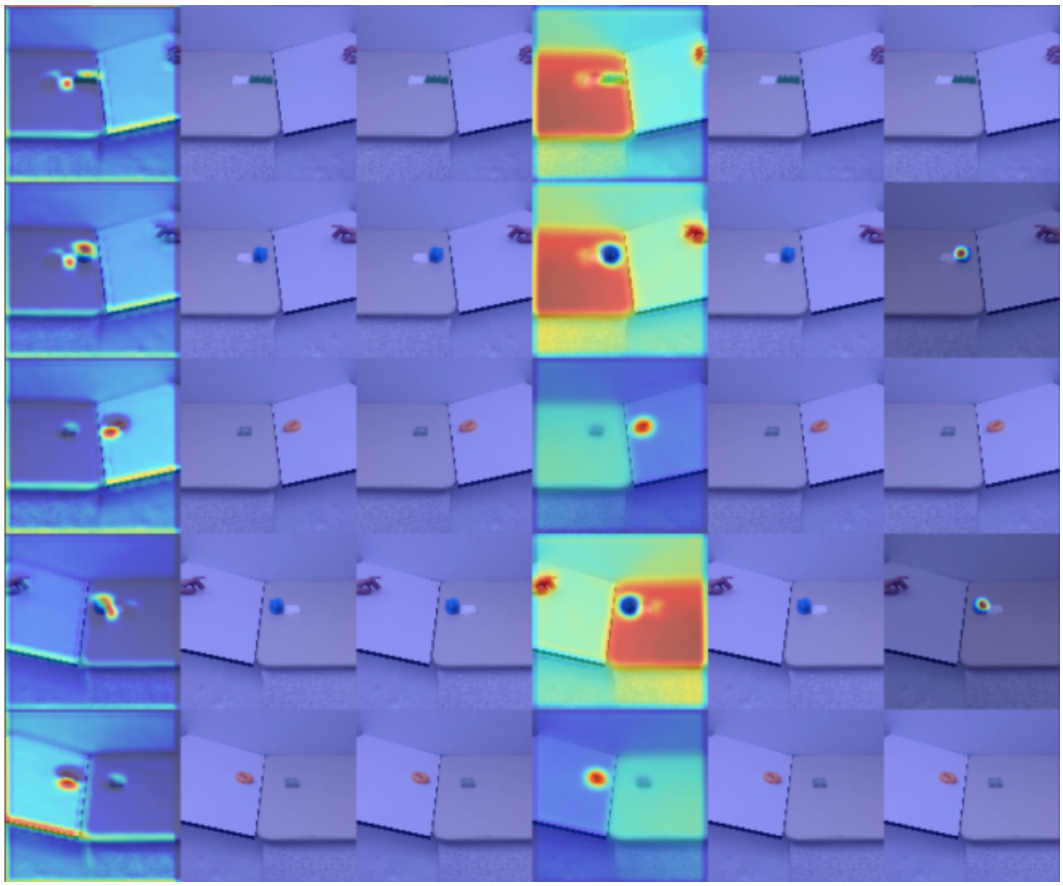
The meta-learned inner loss of the Noether Network is formulated as in equation 2(b), with the learned embedding g_ϕ parameterized as a 2-layer convolutional network where each layer consists of a 5×5 convolution with 32 filters in the first layer and 64 filters in the second layer, a ReLU non-linearity,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

The CNGRAD algorithm of Alet et al. [2] is used to perform tailoring. In the Noether Network, conditional normalization (CN) layers are inserted after each batch 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. We report results obtained with meta-tailoring, where both the base model and the embedding are randomly initialized.

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 400 epochs and approximately 13 hours, and training the Noether Network took approximately 2 days, 23 hours. The error bars reported in figure 4 and figure 6 are standard error of the mean (SEM) for $N = 5$ training seeds.

F Grad-CAM localization maps

In figure 7, Grad-CAM heatmaps are shown for a single time step from each of five example sequences from the test set. The trends discussed in section 4.2 are consistent across these examples as well.



Embedding dimension (after PCA) $\longrightarrow$

Figure 7: Grad-CAM localization maps have consistent behavior across sequences in the test set. Each row corresponds to a different sequence, and each column represents an embedding dimension (first PCA dimension on the left).