SyMetric: Measuring the Quality of Learnt Hamiltonian Dynamics Inferred from Vision

Irina Higgins
DeepMind
London
irinah@deepmind.com

Peter Wirnsberger
DeepMind
London
pewi@deepmind.com

Andrew Jaegle
DeepMind
London
drewjaegle@deepmind.com

Aleksandar Botev
DeepMind
London
botev@deepmind.com

Abstract

A recently proposed class of models attempts to learn latent dynamics from high-dimensional observations, like images, using priors informed by Hamiltonian mechanics. While these models have important potential applications in areas like robotics or autonomous driving, there is currently no good way to evaluate their performance: existing methods primarily rely on image reconstruction quality, which does not always reflect the quality of the learnt latent dynamics. In this work, we empirically highlight the problems with the existing measures and develop a set of new measures, including a binary indicator of whether the underlying Hamiltonian dynamics have been faithfully captured, which we call Symplecticity Metric or *SyMetric*. Our measures take advantage of the known properties of Hamiltonian dynamics and are more discriminative of the model's ability to capture the underlying dynamics than reconstruction error. Using SyMetric, we identify a set of architectural choices that significantly improve the performance of a previously proposed model for inferring latent dynamics from pixels, the Hamiltonian Generative Network (HGN). Unlike the original HGN, the new HGN++ is able to discover an interpretable phase space with physically meaningful latents on some datasets. Furthermore, it is stable for significantly longer rollouts on a diverse range of 13 datasets, producing rollouts of essentially infinite length both forward and backwards in time with no degradation in quality on a subset of the datasets.[1]

1 Introduction

If you want to understand how the world will change, a good place to start is with simple physical principles, such as the laws of motion. One way to achieve good performance in predicting dynamics forward and backward in time is by building a model using the Hamiltonian formalism from classical mechanics. This formalism aims to model physical systems which conserve energy. Apart from real physical problems, other important dynamical systems can also be modelled using this formulation, including GAN optimisation [35, 38, 34, 36], multi-agent learning in non-transitive zero-sum games [5, 6, 53], or the transformations induced by flows in generative models [41]. While many naturally observed dynamics do not preserve energy – for example, energy is often added or dissipated – they can

[1]The code for reproducing all results is available on https://github.com/deepmind/deepmind-research/tree/master/physics_inspired_models.

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

still be modelled using the Hamiltonian formalism by augmenting it with additional terms capturing the deviations from the energy-conserved state [37, 23, 43, 55, 56].

Recently, a body of work has emerged that brings these well-established principles of modelling dynamics from physics – such as the conservation of energy, and numerical formulations from the theory of differential equations – to neural network architectures [49, 20, 9, 3, 58, 8, 4, 31, 50, 43, 56, 12, 17, 24, 14, 32, 11, 45, 57, 54, 48, 21]. Most of these papers, however, learn dynamics directly from an abstract state, such as the positions and momenta of the physical objects. This may be a suitable choice for some applications, for example in molecular simulations, but in situations where inference at test time has to be made from high-dimensional observations, like images – a setup common to many robotics, reinforcement learning or autonomous driving challenges – it is important to be able to have models that can both correctly infer the abstract state space and faithfully capture the underlying Hamiltonian dynamics without access to the ground truth state space information at inference time. Currently only a handful of methods exist for learning dynamics with physical priors from pixels [49, 9, 45, 3, 58]. These models learn in a completely unsupervised manner using pixel observations of the dynamics, however, currently there is no good way to know whether these models have indeed managed to recover the underlying dynamics faithfully through learning.

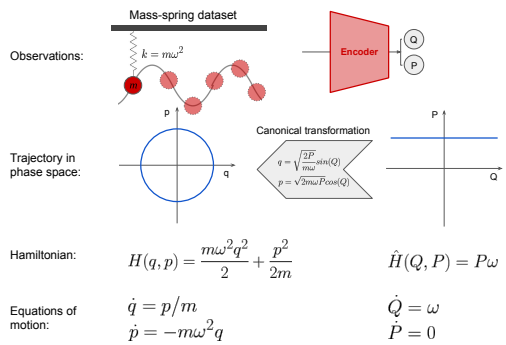

Figure 1: Illustration of a canonical transformation between the Cartesian phase space (q,p) and the action-angle (Q,P) representations for the mass-spring Hamiltonian system. Both coordinate spaces preserve the underlying dynamics.

In this paper we empirically demonstrate that reconstruction based measures typically used in the field are poor indicators of how well the model has learnt the underlying dynamics and propose a better method for evaluating such models. In summary, our contributions are as follows:

1. We introduce performance measures for identifying whether a model trained with the Hamiltonian prior has managed to capture the true dynamics of an energy-conserving system when learning from high-dimensional observations.
2. We use these measures to identify a set of hyperparameters and architectural modifications that significantly improves the performance of Hamiltonian Generative Networks (HGN) [49], an existing state of the art model for recovering Hamiltonian dynamics from pixel observations, both in terms of long time-scale predictions, and interpretability of the learnt latent space.

2 Preliminaries

Hamiltonian dynamics The Hamiltonian formalism is a mathematical formulation of Newton's equations of motion for describing energy-conservative dynamics (see [25, 29]). It describes the continuous time evolution of a system in an abstract phase space with state $\boldsymbol{s} = (\boldsymbol{q}, \boldsymbol{p}) \in \mathbb{R}^{2n}$, where $\boldsymbol{q} \in \mathbb{R}^n$ is a vector of positions, and $\boldsymbol{p} \in \mathbb{R}^n$ is the corresponding vector of momenta. The time evolution of the system in phase space is then given by the Hamiltonian equations of motion:

$$\dot{\boldsymbol{q}} = \frac{\partial \mathcal{H}}{\partial \boldsymbol{p}}, \quad \dot{\boldsymbol{p}} = -\frac{\partial \mathcal{H}}{\partial \boldsymbol{q}} \tag{1}$$

where the Hamiltonian $\mathcal{H} : \mathbb{R}^{2n} \to \mathbb{R}$ maps the state $\boldsymbol{s} = (\boldsymbol{q}, \boldsymbol{p})$ to a scalar representing the energy of the system. Based on the above equations of motion, the energy conservation property is easy to see, since $\frac{dH}{dt} = \frac{\partial H}{\partial q}\dot{\boldsymbol{q}} + \frac{\partial H}{\partial p}\dot{\boldsymbol{p}} = 0$.

Hamiltonian dynamics as a symplectic map The time evolution of Hamiltonian systems is described using a *symplectic map*. To define a symplectic map, we first need to define a symplectic matrix. A matrix M is called symplectic if $M^T A M = A$, where $A = \begin{bmatrix} 0 & \mathbb{I} \\ -\mathbb{I} & 0 \end{bmatrix}$, and $\mathbb{I} \in \mathbb{R}^n$ is the identity matrix. Then, a differentiable map $F(\boldsymbol{s}) : \mathbb{R}^{2n} \to \mathbb{R}^{2n}$ is called symplectic if its Jacobian matrix $J = \frac{\partial F}{\partial \boldsymbol{s}}$

is symplectic everywhere, i.e. $J^T A J = A$. Indeed, that Hamiltonian dynamics are governed by a symplectic map $\frac{d\boldsymbol{s}}{dt} = A\frac{\partial H}{\partial \boldsymbol{s}}$ (see Section 16.3 in [18] for the derivation).

Canonical transformations Physical systems can be equivalently described in any number of arbitrary coordinate systems. For example, the dynamics of a simple mass-spring system can be described equally well using either canonical Cartesian phase space coordinates or the action-angle coordinates (see Fig. 1). A canonical transformation is a mapping that moves between equivalent state spaces in a way that preserves the form of the Hamiltonian dynamics. For example, there is a canonical transformation between the Cartesian phase space coordinates or the action-angle coordinates shown in Fig. 1, since both preserve the correct underlying Hamiltonian dynamics.

More formally, let us consider a Hamiltonian system with a phase-space state $(\boldsymbol{q},\boldsymbol{p})$, a Hamiltonian $\mathcal{H}(\boldsymbol{q},\boldsymbol{p})$ and equations of motion given by Eq. 1. We can transform the state of this system according to new variables[2] $\boldsymbol{Q} = \boldsymbol{Q}(\boldsymbol{q},\boldsymbol{p})$ and $\boldsymbol{P} = \boldsymbol{P}(\boldsymbol{q},\boldsymbol{p})$. In general, such a transformation cannot be expected to preserve the equations of motion in the new variables $(\boldsymbol{Q},\boldsymbol{P})$. However, if there exists a new Hamiltonian $\hat{\mathcal{H}}(\boldsymbol{Q},\boldsymbol{P})$ that describes the same dynamics as the ones described by the original Hamiltonian $\mathcal{H}(\boldsymbol{q},\boldsymbol{p})$, where $\dot{\boldsymbol{Q}} = \frac{\partial \hat{\mathcal{H}}}{\partial \boldsymbol{P}}$ and $\dot{\boldsymbol{P}} = -\frac{\partial \hat{\mathcal{H}}}{\partial \boldsymbol{Q}}$, then such a transformation is called *canonical*. Canonical transformations are also symplectic maps [2]. We provide an illustration of a canonical transformation in Fig. 1 and refer to Stewart [46] for more details.

3 Measuring the quality of learnt Hamiltonian dynamics

To measure whether a model has learnt the underlying Hamiltonian dynamics faithfully, we propose measuring whether there exists a canonical transformation between the ground truth phase space and a subspace or a submanifold within the latent space discovered by the model. Note that a direct comparison between the inferred state space and the ground truth phase space (e.g. L2 distance) is impossible because physical systems can be equivalently described in any number of arbitrary coordinate systems and there is no reason why the models should converge to the phase space arbitrarily chosen as the "ground truth" when learning from pixels.

As discussed in Sec. 2, symplectic maps (or canonical transformations) are typically defined over two spaces of the same dimension. In our case, however, the latent space of the model is most often overparametrised to have more dimensions than the ground truth state. Saying this, if the model has learnt to capture the underlying Hamiltonian dynamics, the dynamics in its latent space should "mimic" the dynamics in the ground truth state space, in the following sense: given a latent space trajectory $\boldsymbol{S}(t) \in \mathbb{R}^{2m}$ and a ground state trajectory $\boldsymbol{s}(t) \in \mathbb{R}^{2n}$ where $m \geq n$ there should exist a map $F : \mathbb{R}^{2m} \to \mathbb{R}^{2n}$ such that the energy landscape defined by the learnt Hamiltonian $\mathcal{H}_m(\boldsymbol{S})$ is a scaled version of the energy landscape defined by the original Hamiltonian $\mathcal{H}_n(F(\boldsymbol{S}))$, i.e. $E_m = \mathcal{H}_m(\boldsymbol{S}) = c\mathcal{H}_n(F(\boldsymbol{S})) = cE_n$ (where c is a constant, and E is the energy of the system, see Sec. A.5 in the Appendix for more motivation for the constant). One can think of the map F as the pseudo-inverse of a symplectic map between the space $\mathbb{R}^{2n}$ of the ground truth state and another space $\mathbb{R}^{2n}$ which is embedded in the higher dimensional space $\mathbb{R}^{2m}$ of the model latent space. This pseudo-inverse preserves the Hamiltonian dynamics of the ground truth state space in the model latent space. If we expand the time derivative of the projection of the learnt state space back to the ground truth state space $\hat{\boldsymbol{s}}(t) = F(\boldsymbol{S}(t))$, we see that

$$
\begin{aligned}
\frac{\partial \hat{\boldsymbol{s}}(t)}{\partial t} &= \frac{\partial F}{\partial \boldsymbol{S}}\frac{\partial \boldsymbol{S}(t)}{\partial t} = \frac{\partial F}{\partial \boldsymbol{S}} A_m \frac{\partial \mathcal{H}_m}{\partial \boldsymbol{S}} \\
&= c\frac{\partial F}{\partial \boldsymbol{S}} A_m \frac{\partial F}{\partial \boldsymbol{S}}^T \frac{\partial \mathcal{H}_n}{\partial \boldsymbol{s}} \\
&= c(J A_m J^T)\frac{\partial \mathcal{H}_n}{\partial \boldsymbol{s}}.
\end{aligned}
\tag{2}
$$

Hence, if we can find a mapping F where $cJA_mJ^T = A_n$ holds, then it suggests the dynamics in the latent space do in fact "mimic" the dynamics in the ground truth state space, and hence the model has recovered the underlying Hamiltonian dynamics of the system.

[2] These can also depend on time but we are not considering time-dependent transformations here. Hence, we will only consider *restricted canonical transformations* in this paper.

To measure how well the model has learnt to mimic the original dynamics, we first learn the mapping $F : \boldsymbol{S} \mapsto \boldsymbol{s}$ that explains as much variance in the original state space $\boldsymbol{s}$ as possible. We then check whether this mapping is symplectic by measuring how much JA_mJ^T deviates from A_n up to a constant scaling factor, where $J = \frac{\partial F}{\partial \boldsymbol{S}}$ is the Jacobian of the learnt mapping, and $A_k = \begin{bmatrix} 0 & \mathbb{I} \\ -\mathbb{I} & 0 \end{bmatrix}$ is a $2k \times 2k$ anti-symmetric matrix. We next discuss each of the two steps in more detail.

Learning to explain the ground truth dynamics We first learn the mapping F between the model state space $\boldsymbol{S}$ and the ground truth phase space $\boldsymbol{s}$. A naive implementation would use an **MLP** to approximate F, however one has to be very careful not to overfit to the training data. In the low data regimes we found that even small MLPs trained with L1 regularization often overfit by learning an unnecessarily complicated map that ends up not being symplectic even if a symplectic map exists (see Sec. A.3 in the Appendix). Hence, we make sure to use at least 1000x more datapoints for training the MLP than the number of its parameters.

Given that learning the map F relies on the availability of ground truth phase space data, which may be hard to obtain in certain domains, we also propose a more data efficient way of learning F that only requires labeling 100 training trajectories. We do so by going back to the basics of machine learning and avoiding overfitting by controlling the expressivity of the mapping F by training a **Lasso regularised linear regression** over a polynomial expansion over the learnt state space features $\boldsymbol{S}$, similarly to DiPietro et al [14]. We start with the expansion order of 1 and progressively increase it up to an upper bound determined by the hyperparameter κ, until enough variance in $\boldsymbol{s}$ is explained. In our experiments we use $\kappa = 5$ (see Alg. 1 in Appendix for more details). To limit the exponential computational cost of computing higher order polynomial expansions of larger latent spaces, where possible we first filter out "uninformative" state dimensions defined as those dimensions that have average KL from the prior less than the threshold $KL[q_\phi(S_i)|p(S_i)] < 0.01$ as in Duan et al [16], however this step is not necessary, and we have successfully applied our approach without this step in practice to a 1024-dimensional latent space.

The MLP and polynomial regression (PR) instantiations of learning the mapping F have different trade-offs. As discussed above, MLP requires orders of magnitude more ground truth state data (in our experiments at least 640x) than PR, however because it is exposed to more data, it can be more accurate than PR, especially in terms of estimating how much variance in the ground truth dynamics can be explained well by the latent dynamics. The MLP can be arbitrarily expressive, while PR calculations get exponentially more expensive as the expressivity of F is increased. PR, however, has the benefit of interpretability.

Having learnt F using either the MLP or PR method, we calculate its **goodness of fit**, R^2, which measures how much information about the ground truth phase space dynamics can be recovered from the learnt state space according to $R^2 = 1 - \frac{\sum(F(\boldsymbol{S}) - \overline{\boldsymbol{s}})^2}{\sum(\boldsymbol{s} - \overline{\boldsymbol{s}})^2}$, where $\overline{\boldsymbol{s}}$ is the mean of the ground truth state variables, $F(\boldsymbol{S})$ are the ground truth state variables predicted from the model state variables $\boldsymbol{S}$, and the summation is over the available data points [13]. R^2 serves as a useful estimate of how much information about the ground truth dynamics is preserved in the latent trajectories. We use it instead of estimation error based alternatives (e.g. MSE, which worked equally well in practice), because it is not sensitive to the magnitude of the data, and hence allows for a more direct comparison between scores obtained from different models trained on different datasets.

Measuring symplecticity of the mapping We check the symplecticity of the learnt mapping F by calculating $\hat{A}_n = JA_mJ^T$. As the next step we could measure directly whether this product is equal to the anti-symmetric matrix A_n, however depending on the model it is possible that the product will instead be equal to A_n^T (we are going to drop the subscript for easier readability where the dimensionality can be inferred from context), which may happen if the Hamiltonian learnt by the model is related to the original Hamiltonian through a negative constant c in Eq. 2 (also see Sec. A.3 in the Appendix). Hence, we instead calculate $\hat{A}\hat{A}^T = JAJ^TJA^TJ^T$ and measure how far it deviates from the identity matrix:

$$Sym = \text{MSE}(c\hat{A}\hat{A}^T, \mathbb{I}) \tag{3}$$

where c is a normalisation factor as per Sec. 3 (see Sec. A.5 in the Appendix for more motivation), and $\mathbb{I} \in \mathbb{R}^{2n}$. Note that c will be different between different models, since it depends on the learnt

Dataset	HGN++					HGN				
	MSE		VPT	Sym	R^2	MSE		VPT	Sym	R^2
	Rec	Ext				Rec	Ext			
Mass-spring	**0.05**	**0.18**	**937.0**	**0.0*/0.0***	**0.99*/1.0***	25.07	197.67	5.75	0.68/0.0	0.88/0.86
Mass-spring +c	**1.63**	**2.77**	**567.5**	**0.03***/0.04*	**0.96*/0.95***	25.49	126.34	8.5	0.78/0.0	0.52/0.5
Pendulum	**1.95**	**10.5**	**85.25**	1.76/0.61	0.54/**0.96**	25.99	294.22	1.75	**0.43**/0.0	**0.79**/0.81
Pendulum +c	26.94	**80.89**	**23.0**	0.5/**0.48**	-0.0/**0.98**	**23.43**	194.52	5.75	**0.37**/50.0	**0.39**/0.55
Matching pennies	**2.25**	**11.71**	**640.5**	**0.35**/0.25	**0.99/0.98**	32.46	562.98	5.75	0.55/**0.18**	0.88/0.95
Rock-paper-scissors	**4.82**	**34.37**	**146.75**	**0.18/0.1**	**0.69/0.96**	138.16	955.31	0.0	0.23/0.12	0.55/0.88
Double pendulum	32.24	**154.83**	**5.5**	**0.28**/0.24	**0.4/0.96**	**26.33**	196.52	2.0	0.3/**0.13**	0.2/0.64
Double pendulum +c	54.7	**87.19**	**5.0**	0.14/0.48	**0.05/0.91**	**26.08**	97.28	2.5	**0.12/0.13**	-0.0/0.6
Two-body	**0.04**	**0.41**	**447.0**	0.22/0.15	**0.91/0.98**	22.33	152.42	12.75	**0.21/0.14**	0.83/**0.98**
Two-body +c	**1.95**	**24.68**	**36.25**	**0.22**/0.19	**0.4/0.95**	25.1	69.88	15.75	0.5/**0.11**	0.32/0.78
3D room - circle	114.43	**370.05**	1.0	**0.11**/max	**0.16/0.97**	**39.55**	1044.6	**1.25**	**0.11**/max	**0.16**/0.35
MD - 4 particles	55.68	**218.47**	0.75	**0.07/0.06**	0.0/**0.94**	**23.41**	1711.71	**4.0**	0.91/**0.06**	0.0/0.82
MD - 16 particles	364.73	447.82	0.0	**0.02/0.02**	0.0/**0.64**	**5.78**	**396.71**	**3.5**	**0.02**/950.0	0.0/0.47

Table 1: The reconstruction (Rec) MSE refers to the MSE measured over the first $T = 60$ timesteps, i.e. the same trajectory length that was used for training, but on test data. The extrapolation (Ext) MSE was computed across the subsequent T timesteps, immediately following those used for training. All MSE values are reported as multiples of 10^7. VPT scores are calculated as the average over forward and backward extrapolation. SyMetric, Sym and R^2 results correspond to MLP/PR implementations. * indicates models with SyMetric $= 1$.

Hamiltonian. We estimate it by finding a constant that minimises Sym for each model separately. Since for symplectic maps the right-hand side in Eq. 3 has to be zero everywhere, we need to evaluate it across all avaliable data. In practice we sample t points across k trajectories to evaluate the Jacobian and average Sym score over. The normalisation factor is calculated as $c = 1/\text{mean}(\max(|\hat{A}\hat{A}^T|))$, where the absolute maximum value of $\hat{A}\hat{A}^T$ is taken at every evaluation point, and their average is taken across all the points from a single trajectory. We calculate a different constant for each trajectory to ensure that on datasets where different trajectories are sampled from different Hamiltonians (e.g. mass-spring +c or pendulum +c), Sym is still meaningful. This does not affect Sym for datasets generated from a single Hamiltonian, where in practice the constant is the same across all trajectories.

3.1 Aggregating into a single indicator: SyMetric

R^2 and Sym measure two orthogonal properties of the learnt dynamics: 1) whether they capture enough information about the ground truth dynamics; and 2) whether the captured dynamics mimic those of the ground truth faithfully. Indeed, these scores can be at the opposite scales of the range in one model. For example, a model may "cheat" and learn to produce perfect trajectories in pixel space by learning $\boldsymbol{q} = [q, \dot{q}, t]$, $\boldsymbol{p} = [0, 0, 0]$, and $\mathcal{H} = p_2$. In this case Hamiltonian dynamics would increase t and, given a powerful enough decoder that can memorise observations indexed by t, such a model would score perfectly on any reconstruction based metric, including VPT. Likewise, given a powerful enough function approximator for F, this model can also have a perfect R^2 score. However, in this case F would not be symplectic, and hence Sym would be high and serve as the only indicator that the model has not actually recovered the underlying Hamiltonian dynamics as hoped. Alternatively, a degenerate F (e.g. the identity function) would be symplectic and result in the perfect $Sym = 0$ score, however in this case if the dynamics learnt by the model do not actually mimic the ground truth dynamics, it will be captured by the low R^2. In other words, if R^2 is low, then the learnt latent space is degenerate and does not preserve all the information about the ground truth dynamics. In this case, the value of Sym is irrelevant and the model has failed to learn well. If R^2 is high but Sym is low, then the learnt dynamics are expressive enough to mimic those of the ground truth, but they are likely to have overfit to the training trajectories and will fail when it comes to extrapolation, since they do not capture the true underlying dynamics. Hence, in order for a model to capture the underlying Hamiltonian dynamics well, the mapping between the latent space and the ground truth phase space needs to be both informative (high R^2) and symplectic (low Sym).

While our general recommendation is for the practitioners to use the R^2 and Sym scores for comprehensive model evaluation, we also propose a way to aggregate the two measures into a single "at-a-glance" score for capturing the main insight: whether a model has discovered the true dynamics or not. We call

this aggregate score the Symplecticity Metric[3] or SyMetric. We define SyMetric to be binary according to

$$\text{SyMetric} = \begin{cases} 1, & \text{if } R^2 > \alpha \wedge Sym < \epsilon \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

We found that SyMetric parameters $\alpha = 0.9$ and $\epsilon = 0.05$ worked well for both MLP and PR implementations.

4 Models

HGN HGN [49] is a generative model that aims to learn Hamiltonian dynamics from pixel observations $\boldsymbol{x}$. It consists of the encoder network that learns to embed a sequence of images $(\boldsymbol{x}_0,...\boldsymbol{x}_T)$ to a lower-dimensional abstract phase space $\boldsymbol{s} \sim q_\phi(\cdot|\boldsymbol{x}_0,...\boldsymbol{x}_T)$ which consists of abstract positions and corresponding momenta $\boldsymbol{s} = (\boldsymbol{q},\boldsymbol{p})$; the Hamiltonian network that maps the inferred phase space to a scalar corresponding to the energy of the system $\mathcal{H}_\gamma(\boldsymbol{s}_t) \in \mathbb{R}$; an integrator that takes in the current state $\boldsymbol{s}_t$, its time derivative calculated using the Hamiltonian network, and the timestep Δt to produce the state $\boldsymbol{s}_{t+\Delta t}$; and the decoder network that maps the position coordinates of the phase space back to the image space $p_\theta(\boldsymbol{x}_t) = d_\theta(\boldsymbol{q}_t)$. HGN is trained using the variational objective

$$\frac{1}{T+1} \sum_{t=0}^{T} \mathbb{E}_{p(\boldsymbol{x})} \left[\mathbb{E}_{q_\phi(\boldsymbol{s}_t|X)} [\log p_\theta(\boldsymbol{x}_t|\boldsymbol{s}_t)] - \mathcal{D}_{KL}(q_\phi(\boldsymbol{s}_t|X)|p(\boldsymbol{s}))\right], \tag{5}$$

where $X = \{\boldsymbol{x}_0,...\boldsymbol{x}_T\}$ and $p(\boldsymbol{s})$ is the isotropic unit Gaussian prior.

HGN++ In order to improve the performance of HGN, we ran a sweep over hyperparameters, investigated architectural changes (see Sec. A.1 in Appendix for more details) and used our proposed new measures described in the Sec. 3 for model selection. We evaluated a range of learning rates, activation functions, kernel sizes for the convolutional layers, hyperparameter settings for the GECO solution for optimising the variational objective [42] and the inference and reconstruction protocols used for training the model. We also considered larger architectural changes, like whether to use the spatial broadcast decoder [52]; whether to use separate networks for inferring the position and momenta coordinates in the encoder; whether to explicitly encourage the rolled out state at time $t+N$ to be similar to the inferred state at time $t+N$ as in [3]; whether to infer the phase space directly or project it through another neural network as in the original HGN work; whether to train the model to predict rollouts forward in time, or both forward and backward in time as in [24]; and finally whether to use a 2D (convolutional) or a 1D (vector) phase-space and corresponding Hamiltonian. We found that a combination of 3×3 kernel sizes and leaky ReLU [33] activations in the encoder and decoder, Swish activations [39] in the Hamiltonian network, as well as a 1D phase-space inferred directly from images used in combination with a Hamiltonian parametrised by an MLP network significantly helped to improve the performance of the model. Furthermore, changing the GECO-based training objective to a β-VAE-based [22] one, training the network for prediction rather than reconstruction and training it explicitly to produce rollouts both forward and backward in time further improved its performance.

5 Datasets

We compare the performance of models on 13 datasets instantiating different types of Hamiltonian dynamics introduced in Botev et al [7], including those of mass-spring, pendulum, double-pendulum and two-body toy physics systems, molecular dynamics, dynamics of camera movements in a 3D environment and learning dynamics in two-player non-transitive zero-sum games known to exhibit Hamiltonian-like cyclic behaviour [5, 6, 53]. All of these (apart from 3D room) were instantiated in two different versions of increasing difficulty. See Sec. A.2 in Appendix for more details.

[3] While we abuse the notation by calling our proposed performance measure a "metric", we could not resist the pun.

6 Baseline measures

A common way to evaluate performance of models of this class is by comparing the mean squared error (MSE) (or equivalent) between the reconstructed and the original pixel observations on a test subset of the data. This has two problems. First, it is well known that pixel reconstruction errors can often be misleading [44, 30, 51, 15] – e.g. low reconstruction errors may be obtained from a perfect reconstruction of the static background while failing to reconstruct the dynamics of interest, if these belong to a relatively small object. Furthermore, each absolute value of the reconstruction error may mean different things across datasets depending on the visual complexity and other particularities of the data, which makes model comparison across datasets hard. Furthermore, good predictive performance on short sequences used for training does not necessarily mean that the model has faithfully captured the underlying dynamics or imply good performance on long time horizon predictions.

Reconstruction and extrapolation errors To demonstrate the problems that arise when using observation-level reconstruction quality for evaluating the model's ability to learn dynamics, we calculate the "reconstruction" pixel MSE – the most commonly used measure of model performance (e.g. employed by [49, 9, 45, 3, 58]), where the model is evaluated on how well it can reproduce the same trajectory length T as was used for training, albeit using test data. We also calculate MSE over extrapolated trajectories, where we continue to roll out the model for a total of $2T$ steps and measure MSE over the last T timesteps, to check whether measuring extrapolation even over short time periods might predict the model's ability to extrapolate further in time better than the "reconstruction" MSE. In all our experiments $T=60$, and for more fair comparison across datasets in all cases MSE is normalised by the average intensity of the ground truth observation as proposed in Zhong et al [57]: $\text{MSE}=||\boldsymbol{x}-\hat{\boldsymbol{x}}||_2^2/||\boldsymbol{x}||_2^2$, where $\boldsymbol{x}_t$ is the ground truth and $\hat{\boldsymbol{x}}_t$ is the reconstructed observation.

Valid Prediction Time We validate our proposed measures by comparing their predictions to another estimate of the model's performance, namely the *Valid Prediction Time* (VPT) (as in Jin et al [26]). VPT is motivated by the intuition that those models that have truly captured the underlying dynamics of energy conserving non-chaotic systems should in principle be able to produce forward and backward rollouts over significantly longer time intervals than those used for training without significant detriment to the reconstruction quality. Note, however, that VPT may also produce false positive or false negative results as will be discussed in Secs. 3 and 7, and hence is not perfect. It is, however, the best alternative measure we could come up with for validating our proposed measures. To calculate VPT we generate a small number of very long trajectories of between 256 and 1000 steps for our datasets (we use 60 step trajectories for training) and measure how long the model's trajectory remains close to the ground truth trajectory in the observation space. It corresponds to the first time step at which the model reconstruction significantly diverges from the ground truth:

$$\text{VPT}=\underset{t}{\operatorname{argmin}}\,[\text{MSE}(\boldsymbol{x}_t,\hat{\boldsymbol{x}}_t)>\lambda]$$

where $\boldsymbol{x}_t$ is the ground truth, $\hat{\boldsymbol{x}}_t$ is the reconstructed observation at time t, and λ is a threshold parameter. For our datasets we found a threshold of $\lambda=0.025$ a reasonable choice based on visual inspection of the rollouts. VPT scores are averaged across 10 trajectories.

7 Results

We first trained different variations of the HGN model as described in Sec. 4 on the mass-spring dataset and found that a particular set of hyperparameters and architectural choices consistently resulted in models that were identified by SyMetric as having learnt the underlying Hamiltonian dynamics faithfully. We refer to this improved version of HGN as HGN++. We then trained this model, as well as the original version of HGN, on all 13 datasets. Table 1 demonstrates that HGN++ is indeed a significant improvement on HGN. While HGN is practically unable to produce good rollout trajectories beyond the 60 timesteps that it was trained on, HGN++ can extrapolate well for at least 267 steps forward and 178 steps backward on average across all datasets (vs 11 and 0 steps by HGN) as evidenced by the VPT scores. Backward extrapolation is achieved by feeding $-\Delta t$ to the integrator at test time. Furthermore, on both the easy and hard (+c) versions of the mass-spring dataset, HGN++ is able to faithfully recover the underlying Hamiltonian dynamics, as identified by the SyMetric score of 1.

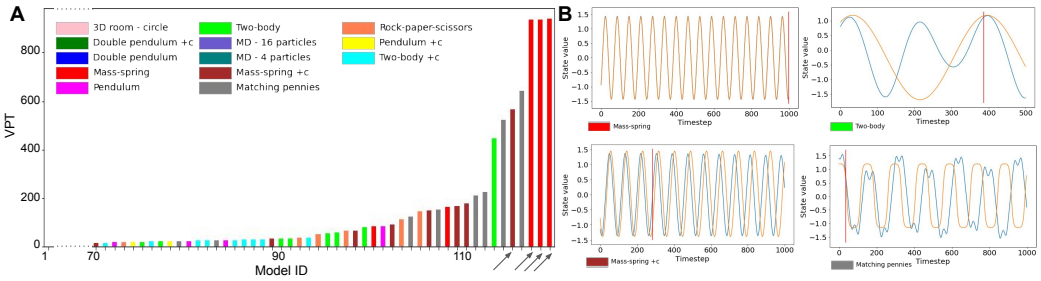

Figure 2: **A**: 119 HGN models trained with different hyperparameters on 13 datasets ordered by average forward and backward VPT score. Arrows point at the models that achieved SyMetric $=1$. **B**: Example trajectories of a single phase space coordinate in the ground truth phase space (orange) and the projection of the model latent space into the ground truth phase space using F (blue). The models highlighted by SyMetric to have discovered the true Hamiltonian dynamics (trained on mass-spring and mass-spring +c datasets) appear to mimic the ground truth dynamics well, while the other models have qualitatively different dynamics despite sometimes scoring similarly by VPT. Red vertical lines indicate the time step where VPT indicates divergence in pixel space.

Given the radically different learning ability of HGN and HGN++, SyMetric is not the only performance measure that differentiates between them. Indeed, MSE results are also indicative of the superior performance of HGN++. However, not all seeds of HGN++ always converge to the same level of performance, and indeed slight modifications in hyperparameters often affect the model's learning ability in ways that have a negligible effect on MSE, yet can drastically affect whether the models are able to uncover the true Hamiltonian dynamics or not. When we look at the top 20% of model seeds in terms of their VPT performance across all datasets, and compare the different measures in terms of how well they differentiate between the better and worse performing seeds based on how well they correlate with VPT (averaged forward and backward), reconstruction MSE has 0.07 absolute Spearman correlation, extrapolation MSE is better at 0.33, however R^2 has stronger correlation of 0.59(0.41) and Sym has correlation of 0.37(0.45) MLP(PR). Hence, when it comes to doing more precise model or hyperparameter selection and differentiating between models that are not obviously failing, it appears that the currently widely used reconstruction MSE measure is very poor, and using either R^2 or Sym is more informative.

Fig. 2A orders the 119 models in terms of their average VPT scores. We see that the mass-spring dataset is clearly solved, with a number of HGN++ seeds with slightly different hyperparameters achieving close to perfect 1000-step extrapolations both forward and backward in time after being trained on just 60 steps. The figure also demonstrates that all of these models are picked out by SyMetric as having discovered the true Hamiltonian of the system. However, at the next step change down in VPT measures, we see inconsistent predictions from VPT and SyMetric. A number of model seeds trained on the matching pennies and two-body datasets are scored highly by VPT, but are not highlighted by SyMetric to have discovered the true Hamiltonian, while a HGN++ seed trained on the mass-spring +c data with similar VPT is highlighted as having done so. Which measure is correct: VPT or SyMetric? When we visualise an example trajectory from these models, mapping the learnt latent space to the ground truth through F and visualising both (see Fig. 2B), we see that SyMetric is in fact correct. The latent dynamics of models trained on matching pennies and two-body datasets are qualitatively different from those exhibited by the ground truth system, while the dynamics from the model trained on mass-spring +c appear to be mimicking those of the ground truth.

In order to further verify the validity of SyMetric, Fig. 3A-B visualises four trajectories in the ground truth phase space and in the latent space learnt by the HGN++ seed highlighted as have learnt the underlying Hamiltonian dynamics on the mass-spring +c dataset. It is clear that latent Q_3 and its corresponding P_3 were able to capture the ground truth state dynamics well. Indeed, when we visualise the extrapolated rollouts of the four sampled trajectories Fig. 3C, we see that they still look good even after 1000 steps, thus verifying that the model is indeed able to produce good rollouts and has appeared to have recovered the underlying ground truth system well. So what causes the relatively low VPT scores recorded for this model? It appears that this discrepancy arises due to a slight error in the estimation of the initial conditions by the model. When we visualise the 1000-step rollout in the ground truth state against the projection of the model's latent state into the ground truth phase space, it is clear

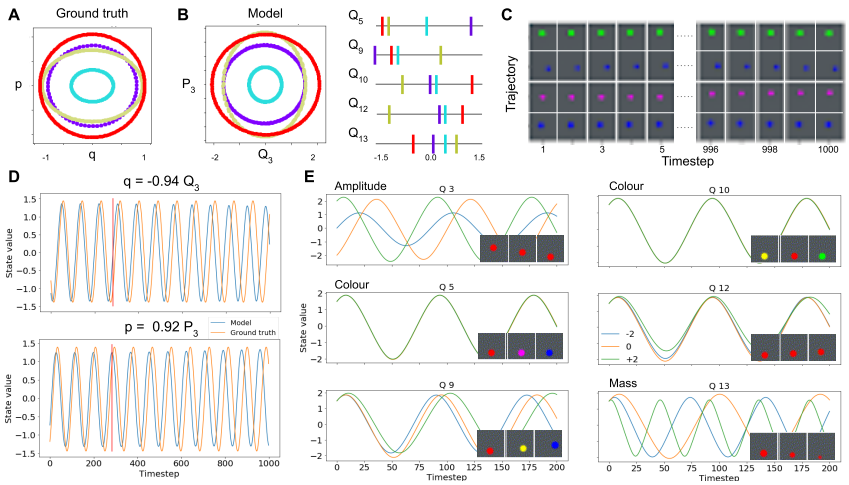

Figure 3: **A**: Visualisation of four trajectories of the mass-spring +c dataset in the ground truth phase space. **B**: Same trajectories visualised in the informative latents of a trained HGN++ with the perfect SyMetric score. If both the Q_i and its corresponding P_i are informative, the trajectories are plotted as a scatter plot. Otherwise they are plotted as points in the corresponding latent dimension. **C**: 1000 step rollouts produced by the model in B of the four trajectories in A-B. The reconstructions are of good quality even after 1000 steps. **D**: Same 1000 step rollouts as in C but shown in the ground truth phase space and the projection of HGN++ from B into the ground truth phase space as produced by the regression step of SyMetric. Due to slight errors in the inference of the initial state, the trajectories diverge after about 300 steps, resulting in a suboptimal VPT score. **E**: Latent traversals of the informative dimensions of HGN++ shown in B. Each plot shows a 200 step rollout trajectory of Q_3 when setting the value of the traversed latent dimension to -2, 0 or 2. The inset images demonstrate the corresponding effect on the reconstruction of the first frame of the trajectory (from left to right). Latents Q_3, Q_5, Q_{10} and Q_{13} appear to have an interpretable meaning, which is indicated in the left top corner of the corresponding subplot.

that the two trajectories are slightly out of phase with each other, which results in a divergence around the time step captured by the VPT score (indicated by the red vertical line in Fig. 3D). Hence, although the model was able to capture the underlying Hamiltonian faithfully as captured by SyMetric score, it appears that its VPT scores were handicapped by the slight errors in the inferred initial conditions.

Furthermore, the HGN++ shown in Fig. 3 had seven informative dimensions at the end of training. Two of them – Q_3 and P_3 – ended up learning the dynamics that mimic the ground truth phase space. The other 5 latents did not have an informative momentum component, indicating that the model used these latents as constants. When we investigated what these latents learnt to represent by traversing their values and visualising the resulting effect on the dynamics in Q_3, we notice that many of these dimensions are physically meaningful – Q_5 and Q_{10} learnt to represent variations in colour, while Q_{13} learnt to represent the mass parameter of the Hamiltonian (see relevant equation in Appendix Tbl. 2). Hence, it appears that HGN++ was able to address another criticism of the original HGN [9], i.e. lack of interpretability of its learnt latent space.

8 Conclusions

In this paper we have introduced a set of novel performance measures – R^2, Sym and their binary combination, SyMetric, which serve as better proxies than the existing measures of whether a model with a Hamiltonian prior has captured the underlying dynamics of a Hamiltonian system from high-dimensional observations, such as pixel images. We have validated our measures both quantitatively – by comparing them to the VPT score, another proxy of the models performance which measures the ability of the model to produce faithful trajectories in extreme extrapolation regimes in the high-dimensional observation space; and qualitatively – by visualising the latent space of the model. We have shown that we were able to use SyMetric to identify a set of hyperparameters and architectural modifications that improved the previously published HGN model in terms of extrapolation performance and interpretability of its latent space, in effect "solving" 2/13 datasets considered here. We

hope that SyMetric can be used by the community to make further progress towards building models with Hamiltonian priors that can learn dynamics from pixels.

While SyMetric requires ground truth phase space information in addition to pixel observations, this information is only required for a relatively small number of trajectories (we used 100 in the PR case), and for relatively short trajectory lengths (60 steps, as was used for training) – the amount of data that would not be sufficient to train the models. This is in contrast to the significantly longer trajectories (256-1000 steps) that were necessary to compute VPT scores – the only observation-space measure to achieve comparable results. By avoiding the need to generate long trajectories, we hope that SyMetric can be a good option for evaluating models that learn from observation where collecting such long trajectories is expensive, e.g. due to the computational costs of running a simulation or other costs corresponding to collecting experimental data.

SyMetric has the drawback of relying on multiple measures: R^2 and Sym. As part of future work we hope to replace this with a single step by using function approximators for F that are symplectic by design (e.g. 26, 40), thus producing a single measure equivalent to R^2, which will only be high if there exists a symplectic mapping between a subspace of the latent space of the model and the ground truth phase space. Furthermore, we hope to extend SyMetric to models that incorporate the Lagrangian prior in their dynamics (e.g. [45, 3, 58]), which should be possible due to the equivalence of the Hamiltonian and the Lagrangian formalisms, and the ability to map between them through the Legendre transform.

Finally, it would be interesting to investigate in detail how SyMetric compares to VPT on systems that are known to exhibit chaotic dynamics (e.g. the MD datasets). This type of problem poses a significant challenge to our models because tiny perturbations to the initial conditions grow exponentially causing trajectories to diverge within a few hundred training steps (e.g. see [19]). While this poses a fundamental challenge to our training procedure and all evaluations metrics, we believe that the significantly shorter trajectories required for SyMetric may be advantageous.

References

[1] J. L. F. Abascal and C. Vega. A general purpose model for the condensed phases of water: Tip4p/2005. *J. Chem. Phys.*, 123:234505, 2005.

[2] R. Abraham and J.E. Marsden. *Foundations of Mechanics*. AMS Chelsea publishing. AMS Chelsea Pub./American Mathematical Society, 2008.

[3] Christine Allen-Blanchette, Sushant Veer, Anirudha Majumdar, and Naomi Ehrich Leonard. LagNetViP: A Lagrangian neural network for video prediction. In *Proceedings of AAAI Conference on Artificial Intelligence*, 2020.

[4] Omri Azencot, N Benjamin Erichson, Vanessa Lin, and Michael W Mahoney. Forecasting sequential data using consistent Koopman autoencoders. In *Proceedings of International Conference on Machine Learning (ICML)*, 2020.

[5] James P Bailey and Georgios Piliouras. Multi-agent learning in network zero-sum games is a Hamiltonian system. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2019.

[6] David Balduzzi, Sebastien Racaniere, James Martens, Jakob Foerster, Karl Tuyls, and Thore Graepel. The mechanics of n-player differentiable games. In *International Conference on Machine Learning*, pages 354–363, 2018.

[7] Alexandar Botev, Andrew Jaegle, Peter Wirnsberger, Daniel Hennes, and Irina Higgins. Which priors matter? Benchmarking models for learning latent dynamics. In *Proceedings of Neural Information Processing Systems (NeurIPS)*, 2021.

[8] Zhengdao Chen, Jianyu Zhang, Martin Arjovsky, and Léon Bottou. Symplectic recurrent neural networks. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2020.

[9] Anshul Choudhary, John F Lindner, Elliott G Holliday, Scott T Miller, Sudeshna Sinha, and William L Ditto. Forecasting Hamiltonian dynamics without canonical coordinates. *Nonlinear Dynamics*, pages 1–10, 2020.

[10] Wendy D. Cornell, Piotr Cieplak, Christopher I. Bayly, Ian R. Gould, Kenneth M. Merz, David M. Ferguson, David C. Spellmeyer, Thomas Fox, James W. Caldwell, and Peter A. Kollman. A second generation force field for the simulation of proteins, nucleic acids, and organic molecules. *J. Am. Chem. Soc.*, 117(19):5179–5197, 1995.

[11] Miles Cranmer, Sam Greydanus, Stephan Hoyer, Peter Battaglia, David Spergel, and Shirley Ho. Lagrangian neural networks. In *ICLR Deep Differential Equations Workshop*, 2020.

[12] Shaan A. Desai, Marios Mattheakis, and Stephen J. Roberts. Variational integrator graph networks for learning energy-conserving dynamical systems. *Physical Review E.*, 104(3), 2020.

[13] Jay L. Devore. *Probability and Statistics for Engineering and the Sciences*. Spinger, 2008.

[14] Daniel M. DiPietro, Shiying Xiong, and Bo Zhu. Sparse symplectically integrated neural networks. In *Proceedings of Neural Information Processing Systems (NeurIPS)*, 2020.

[15] Alexey Dosovitskiy and Thomas Brox. Generating images with perceptual similarity metrics based on deep networks. In *Proceedings of Neural Information Processing Systems (NeurIPS)*, 2016.

[16] Sunny Duan, Loic Matthey, Andre Saraiva, Nicholas Watters, Christopher P Burgess, Alexander Lerchner, and Irina Higgins. Unsupervised model selection for variational disentangled representation learning. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2020.

[17] Marc Finzi, Ke Alexander Wang, and Andrew Gordon Wilson. Simplifying Hamiltonian and Lagrangian neural networks via explicit constraints. In *Proceedings of Neural Information Processing Systems (NeurIPS)*, 2020.

[18] Jason Frank. Symplectic flows and maps and volume preservation. https://webspace.science.uu.nl/ frank011/Classes/numwisk/ch16.pdf, 2008.

[19] Daan Frenkel and Berend Smit. *Understanding Molecular Simulation*. Academic Press, San Diego, second edition, 2002.

[20] Samuel Greydanus, Misko Dzamba, and Jason Yosinski. Hamiltonian neural networks. In *Proceedings of Neural Information Processing Systems (NeurIPS)*, 2019.

[21] Niklas Heim, Václav Šmídl, and Tomáš Pevnỳ. Rodent: Relevance determination in differential equations. *arXiv preprint arXiv:1912.00656*, 2019.

[22] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. β-VAE: Learning basic visual concepts with a constrained variational framework. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2017.

[23] William G. Hoover. Canonical dynamics: Equilibrium phase-space distributions. *Physical Review A*, 31:1695–1697, 1985.

[24] In Huh, Eunho Yang, Sung Ju Hwang, and Jinwoo Shin. Time-reversal symmetric ODE network. In *Proceedings of Neural Information Processing Systems (NeurIPS)*, 2020.

[25] Maxim Jeffs. Harvard, Lecture Notes: Classical mechanics and symplectic geometry, 2020. URL: `http://people.math.harvard.edu/~jeffs/SymplecticNotes.pdf`.

[26] Pengzhan Jin, Zhen Zhang, Aiqing Zhu, Yifa Tang, and George Em Karniadakis. SympNets: Intrinsic structure-preserving symplectic networks for identifying Hamiltonian systems. *Neural Networks*, 132:166–179, 2020.

[27] J. E. Jones and Sydney Chapman. On the determination of molecular fields. –II. From the equation of state of a gas. *Proc. R. Soc. Lond. A*, 106(738):463–477, 1924.

[28] Rishabh Kabra, Chris Burgess, Loic Matthey, Raphael Lopez Kaufman, Klaus Greff, Malcolm Reynolds, and Alexander Lerchner. Multi-object datasets. https://github.com/deepmind/multi-object-datasets/, 2019.

[29] Lev Davidovich Landau and Evgenii Mikhailovich Lifshitz. *Course of theoretical physics*. Elsevier, 2013.

[30] Anders Boesen Lindbo Larsen, Søren Kaae Sønderby, Hugo Larochelle, and Ole Winther. Autoencoding beyond pixels using a learned similarity metric. In *Proceedings of International Conference on Machine Learning (ICML)*, 2016.

[31] Yunzhu Li, Toru Lin, Kexin Yi, Daniel Bear, Daniel LK Yamins, Jiajun Wu, Joshua B Tenenbaum, and Antonio Torralba. Visual grounding of learned physical models. In *Proceedings of International Conference on Machine Learning (ICML)*, 2020.

[32] Michael Lutter, Christian Ritter, and Jan Peters. Deep Lagrangian networks: Using physics as model prior for deep learning. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2019.

[33] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proceedings of International Conference on Machine Learning (ICML)*, 2013.

[34] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for GANs do actually converge? In *Proceedings of International Conference on Machine Learning (ICML)*, 2018.

[35] Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. The numerics of GANs. In *Proceedings of Neural Information Processing Systems (NeurIPS)*, 2017.

[36] Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2017.

[37] Shuichi Nosé. A unified formulation of the constant temperature molecular dynamics methods. *J. Chem. Phys.*, 81(1):511–519, 1984.

[38] Chongli Qin, Yan Wu, Jost Tobias Springenberg, Andrew Brock, Jeff Donahue, Timothy P Lillicrap, and Pushmeet Kohli. Training generative adversarial networks by solving ordinary differential equations. In *Proceedings of Neural Information Processing Systems (NeurIPS)*, 2020.

[39] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Swish: a self-gated activation function. *arXiv preprint arXiv:1710.05941*.

[40] Katharina Rath, Christopher G. Albert, Bernd Bischl, and Udo von Toussaint. Symplectic Gaussian process regression of maps in Hamiltonian systems. *Chaos*, 31, 2021.

[41] Danilo Jimenez Rezende, Sébastien Racanière, Irina Higgins, and Peter Toth. Equivariant Hamiltonian flows. In *NeurIPS workshop: Machine Learning and the Physical Sciences*, 2019.

[42] Danilo Jimenez Rezende and Fabio Viola. Taming VAEs. *arXiv preprint arXiv:1810.00597*, 2018.

[43] Manuel A Roehrl, Thomas A Runkler, Veronika Brandtstetter, Michel Tokic, and Stefan Obermayer. Modeling system dynamics with physics-informed neural networks based on Lagrangian mechanics. In *International Federation of Automatic Control (IFAC) World Congress*, 2020.

[44] Oleh Rybkin. The reasonable ineffectiveness of pixel metrics for future prediction (and what to do about it), 2018. URL: `https://bit.ly/2YHPRg2`.

[45] Steindor Saemundsson, Alexander Terenin, Katja Hofmann, and Marc Deisenroth. Variational integrator networks for physically structured embeddings. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2020.

[46] Iain W. Stewart. MIT, Lecture Notes: Advanced Classical Mechanics, 2016. URL: `https://bit.ly/3oP7fuc`.

[47] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2012.

[48] Yunjin Tong, Shiying Xiong, Xingzhe He, Guanghan Pan, and Bo Zhu. Symplectic neural networks in Taylor series form for Hamiltonian systems. *Journal of Computational Physics*, 437, 2020.

[49] Peter Toth, Danilo Jimenez Rezende, Andrew Jaegle, Sébastien Racanière, Aleksandar Botev, and Irina Higgins. Hamiltonian generative networks. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2020.

[50] Silviu-Marian Udrescu and Max Tegmark. Symbolic pregression: Discovering physical laws from raw distorted video. *Physical Review E*, 103, 2021.

[51] David Warde-Farley and Yoshua Bengio. Improving generative adversarial networks with denoising feature matching. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2017.

[52] Nicholas Watters, Loic Matthey, Christopher P Burgess, and Alexander Lerchner. Spatial broadcast decoder: A simple architecture for learning disentangled representations in VAEs. *arXiv preprint arXiv:1901.07017*, 2019.

[53] Jinshan Wu. Hamiltonian formalism of game theory. *arXiv preprint quant-ph/0501088*, 2005.

[54] Shiying Xiong, Yunjin Tong, Xingzhe He, Cheng Yang, Shuqi Yang, and Bo Zhu. Nonseparable symplectic neural networks. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2021.

[55] Yaofeng Desmond Zhong, Biswadip Dey, and Amit Chakraborty. Dissipative SymODENet: Encoding Hamiltonian dynamics with dissipation and control into deep learning. *arXiv preprint arXiv:2002.08860*, 2020.

[56] Yaofeng Desmond Zhong, Biswadip Dey, and Amit Chakraborty. Symplectic ODE-net: Learning Hamiltonian dynamics with control. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2020.

[57] Yaofeng Desmond Zhong, Biswadip Dey, and Amit Chakraborty. Benchmarking energy-conserving neural networks for learning dynamics from data. In *Conference on Learning for Dynamics and Control (L4DC)*, 2021.

[58] Yaofeng Desmond Zhong and Naomi Leonard. Unsupervised learning of Lagrangian dynamics from images for prediction and control. *Proceedings of Neural Information Processing Systems (NeurIPS)*, 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] See Sec. 8
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See zip archive in supplementary materials
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Both in main text and Appendix
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] Instead we provide the best seed results and a full breakdown of performance by model seed as it is more appropriate to showcase our proposed measure of model performance
   (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] See Appendix Sec. A.1
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] See Sec. 5
   (b) Did you mention the license of the assets? [Yes] See Appendix Sec. A.2
   (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 Appendix

A.1 HGN++

The list of hyperparameters and architectural changes investigated to improve the performance of HGN is listed below. All models were run on P100 GPUs, with one GPU per learner, and each type of evaluator: for calculating test reconstruction losses, SyMetric with PR, SyMetric with MLP and VPT. We used an internal cluster to run the models, and each model was trained for 500k steps.

Training scheme: The original HGN was using a sequence of 30 images $(\boldsymbol{x}_1,...\boldsymbol{x}_{30})$ to infer the latent representation $\boldsymbol{z}_1$. The model would then produce a 30 step rollout and the reconstruction error used for training would be calculated between the original data used for inference and the reconstructed $(\hat{\boldsymbol{x}}_1,...\hat{\boldsymbol{x}}_{30})$. Instead we investigated whether the model would work better if the original sequence of images $(\boldsymbol{x}_1,...\boldsymbol{x}_T)$ was used to infer the state at time $\boldsymbol{z}_T$ rather than $\boldsymbol{z}_1$, so that the rollouts used for training are into the future, and the reconstructed frames are not the ones seen by the encoder model. We found that this latter setup worked better for HGN++.

We further investigated how many steps was required for inference: [5-30]; and how many steps the model should reconstruct: 30, 60, 90. We found that using 5 inference steps and 60 reconstruction steps worked well for HGN++.

Furthermore, we investigated whether it is important to train the model to be able to explicitly produce good rollouts backwards in time (rather than just forward) as per **(author?)** [24]. We found that the forward/backward straining scheme improved the performance of HGN++.

Encoder type: The original HGN was using a single encoder network to predict the final state $(\boldsymbol{q},\boldsymbol{p})$. We also investigated the value of other kinds of encoders: 1) separate networks for encoding $\boldsymbol{q}(\boldsymbol{x}_1,...\boldsymbol{x}_T)$ and $\boldsymbol{p}(\boldsymbol{x}_1,...\boldsymbol{x}_T)$; 2) stacked encoder, where the single images are used to infer the position $\boldsymbol{q}(\boldsymbol{x}_i)$, and the inferred position vectors are stacked to infer the momenta $\boldsymbol{p}(\boldsymbol{q}_1,...\boldsymbol{q}_T)$. We found that the original encoder worked the best for HGN++.

The original HGN used an additional MLP to map between the latent sample inferred by the encoder network $\boldsymbol{z} \sim q_\phi(\cdot|\boldsymbol{x}_1,...\boldsymbol{x}_T)$, and the state $\boldsymbol{s} = (\boldsymbol{q},\boldsymbol{p})$ that was unrolled through the Hamiltonian equations of motion according to $\boldsymbol{s} = \text{MLP}(\boldsymbol{z})$. We found that removing this additional transformation and using the encoder sample directly as state worked better for HGN++.

Decoder type: We investigated whether replacing a ResNet based decoder with an MLP or a Spatial Broadcast Decoder [52] might be beneficial for HGN. We found that the original ResNet was the best choice.

Matching inferred states with rolled out states: similarly to **(author?)** [3] to investigated whether introducing an extra term to the objective function shown in Eq. 5 used to train HGN helps imrove its performance. The extra term forces the state at time step i produced through integrating the equations of motion $\hat{\boldsymbol{s}}_i$ to match the state inferred by the encoder $\boldsymbol{s}_i \sim q_\phi(\cdot|\boldsymbol{x}_{i-T},...\boldsymbol{x}_i)$. We tried different weighing of this additional terms with respect to the original objective function: $\mathcal{L}_{HGN} + \lambda||\hat{\boldsymbol{s}}_i - \boldsymbol{s}_i||_2^2$, with $\lambda \in \{0.01,\ 0.1,\ 1,\ 10\}$. We also tried enforcing this extra constraint every $N \in \{6,\ 10,\ \mathbb{U}(T)\}$ steps, where $\mathbb{U}(T)$ stands for uniform sampling of N for each trajectory between 1 and T. We found that none of the choices made a significant difference to HGN training.

Loss regulariser: In the original HGN implementation, the variational objective in Eq. 5 was optimised using the GECO solution [42]. We ran a hyperparameter sweep for $\kappa \in [1e-6,\ 0.01]$. We also considered another option for optimising the variational objective – β-VAE [22], with a hyperaparameter sweep over $\beta \in [0,\ 20]$. We found that β-VAE objective significantly outperformed the GECO one, and β around 0.1 or 1 tended to work best for HGN++.

Latent space: In the original HGN implementation, the inferred state was 3D: $\boldsymbol{s} \in \mathbb{R}^{4\times4\times32}$. Here we investigated whether a 1D vector based latent state would perform better, e.g. $\boldsymbol{s} \in \mathbb{R}^{32}$. We tried different options for aggregating the original 3D representation into a 1D one: using an MLP, through spatial average or max operation, or through a linear projection. The latter worked the best for HGN++. We also investigated the size of the latent representation and found that 32 dimensions worked as well as the larger options.

Hamiltonian network: We investigated whether replacing the Hamiltonian network from a convolutional architecture to an MLP might be beneficial. We tried 1-4 layer MLPs with hidden layer of size

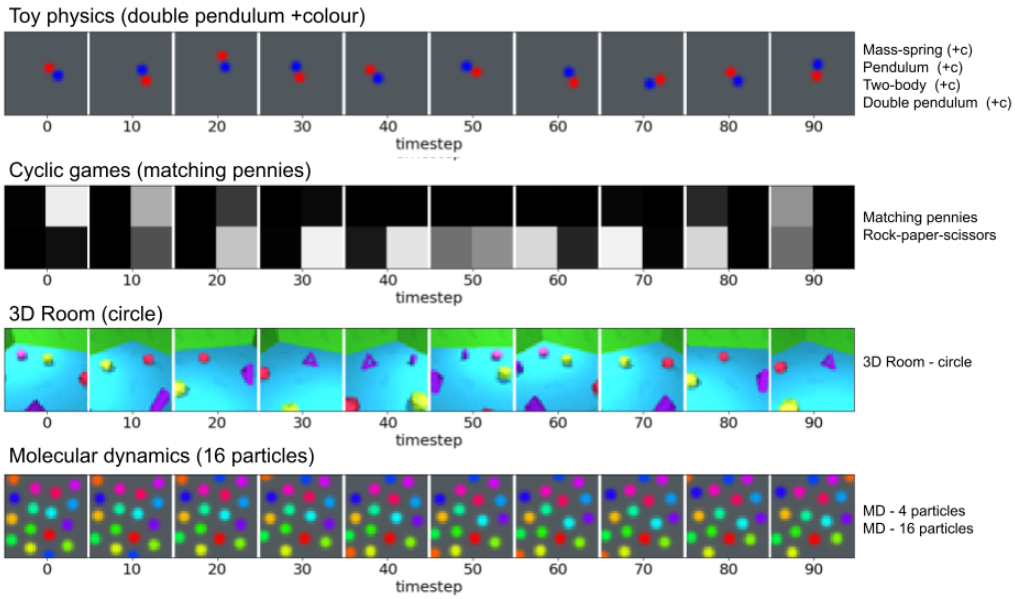

Figure 4: Visualisation of observations from a subset of datasets used in this paper.

250, and found that 4 layers gave us the best performance in HGN++. We also tried adding L1, L2 or a weighted mix of both to the Hamiltonian weights, but found that it did not help. Exchanging Softplus activations with Swish activations on the other hand made a difference.

Activation function: We investigated the role of ReLU, leaky ReLU, Softplus and Swish activations in the encoder and decoder of HGN, and found the leaky ReLU worked the best.

Integrators: We found taht the original leap-frog integrator worked better than the others considered (Euler, RK4). We found that the original $\Delta t = 0.125$ worked better than other choices (e.g. 0.25).

Training steps: The original HGN was trained over 15k steps, We found that increasing the number of steps up to 1mln resulted in better performance. In all reported experiments we used 500k training steps.

Learning rate: We found that the original learning rate of 1.5e-4 worked well for our experiments.

Batch size: We found that the models were training well with any batch size considered: 10, 32, or 128. The reported experiments used batch size 128.

A.2 Datasets

All datasets have $32 \times 32 \times 3$ observations. The ground truth phase space is 2D for mass-spring, pendulum and 3D room datasets, 4D for the matching pennies, double pendulum and two-body datasets, 6D for the rock-paper-scissors dataset, 8D for the 4-particle MD dataset, and 32D for the 16-particle MD dataset. Full description of the datasets can be found in Botev et al [**?**]. A visualisation of example datasets from each type is shown in Fig. 4. All datasets are available under the Apache V2 license here: `https://github.com/deepmind/dm_hamiltonian_dynamics_suite`.

Toy physics We re-use the mass-spring, pendulum and two-body datasets used in the original work on building models with Hamiltonian priors [20, 49], as well as a new double-pendulum dataset. The dynamics in these systems are governed by the Hamiltonians shown in Tbl. 2. The initial conditions for each dataset are sampled in the following way:

- Mass Spring - we sample q and p together from the uniform distribution over the annulus with lower radius bound 0.1 and upper radius bound 1.0 and the we multiply p by $\sqrt{km}$.
- Pendulum - we sample q and p together from the uniform distribution over the annulus with lower radius bound 1.3 and upper radius bound 2.3.
- Double Pendulum - we sample the states of both pendulums analogously to Pendulum.

- Two Body Problem - we follow the same protocol as in [49].

The sampling distribution for every dataset and parameter are shown in Table 3, where k,l,m and g are the parameters corresponding to the spring force coefficient of the mass-spring system, pendulum length, mass and gravitational force respectively. In the original datasets these values were fixed. We also generated more challenging versions of these datasets, where these parameters are sampled. We also randomly sample the horizontal positions and colours of the masses in both datasets. Although the latter do not affect the resulting dynamics, we wanted to test whether the models are capable of learning in this regime. We denote the more challenging versions of the datasets as "+c". For these datasets the radius of the rendered ball is proportional to its mass (see Fig. 4, top).

Dataset	Hamiltonian $\mathcal{H}(q,p)$	Hyperparameters
Mass Spring	$k\frac{q^2}{2}+\frac{p^2}{2m}$	k,m
Pendulum	$mlg(1-\cos(q))+\frac{p^2}{2lm}$	m,l,g
Double Pendulum	$\frac{m_2 l_2^2 p_1^2+(m_1+m_2)l_1^2 p_2^2-2m_2 l_1 l_2 p_1 p_2 cos(q_1-q_2)}{2m_2 l_1^2 l_2^2(m_1+m_2 sin(q_1-q_2)^2}-(m_1+m_2)gl_1 cos(q_1)-m_2 g l_2 cos(q_2)$	m_1,m_2,l_1,l_2,g
N-Body Problem	$-\sum_{i<j}\frac{gm_i m_j}{\|\|q_i-q_j\|\|}+\sum_i\frac{\|\|p_i\|\|_2^2}{2m_i}$	$g,m_1,\ldots,m_n$

Table 2: The Hamiltonians used for simulating all of the classical mechanics systems.

Dataset	Hyperparameters
Mass Spring	$k=2.0$ $m\sim U(0.2,1.0)$
Pendulum	$m\sim U(0.5,1.5)$ $g\sim U(3.0,4.0)$ $l\sim U(0.5,1.0)$
Double Pendulum	$m\sim U(0.4,0.6)$ $g\sim U(2.5,4.0)$ $l\sim U(0.75,1.0)$
Two Body	$m\sim U(0.5,1.5)$ $h\sim U(0.5,1.5)$

Table 3: Sampling protocol for the hyperparameters of the coloured Toy physics datasets.

Cyclic games The Multi-agent cyclic games dataset are generating by using the continuous-time two population replicator dynamics are defined as:

$$\begin{aligned}\dot{x}_i &= x_i\left[(\boldsymbol{A}\boldsymbol{y})_i-\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{y}\right]\\ \dot{y}_j &= y_j\left[\left(\boldsymbol{x}^T\boldsymbol{B}\right)_j-\boldsymbol{x}^T\boldsymbol{B}\boldsymbol{y}\right]\end{aligned} \tag{6}$$

where, $\boldsymbol{A}=-\boldsymbol{B}$ are the payoff matrices of a zero-sum game for the row and column player respectively, and $(\boldsymbol{x},\boldsymbol{y})$ the joint strategy profile. We generate ground-truth trajectories by integrating the coupled set of ODEs equation 6 using an improved Euler scheme or RK45. In both cases the ground-truth state, i.e., joint strategy profile (joint policy), and its first order time derivative, is recorded at regular time intervals Δt. Trajectories start from uniformly sampled points on the product of the policy simplexes. No noise is added to the trajectories.

Molecular Dynamics The goal of this set of datasets is to benchmark the performance of models on problems involving complex many-body interactions. In particular, the generated datasets employ a Lennard-Jones (LJ) potential [27], which is a popular benchmark problem and an integral part of more complex force fields used, for example, to model water [1] or proteins [10].The two generated LJ datasets have increasing complexity: one comprising only 4 particles at a very low density and another one for a 16-particle liquid at a higher density. These MD datasets are rendered using the same scheme as the toy physics datasets. All masses are set to unity and particles are represented by circles of equal size with a radius value adjusted to fit the canvas well. In addition, different colours are assigned to the particles to facilitate tracking their trajectories.

3D Room To evaluate the ability of models to deal with complex 3D visuals, we use a dataset of MuJoCo [47] scenes consisting of a camera moving around a room with 5 randomly placed objects. The objects are sampled from four shape types: a sphere, a capsule, a cylinder and a box. Each room was different due to the randomly sampled colours of the wall, floor and objects similar to [28]. The dynamics are created by rotating the camera around a single randomly sampled parallel of the unit hemisphere centered around the center of the room. The rendered scenes are used as observations, and the Cartesian coordinates of the camera and its velocities estimated through finite differences are used as the state.

A.3 SyMetric

A.3.1 Jacobian calculations

We have a Jacobian

$$J = \begin{bmatrix} \frac{\partial q}{\partial Q} & \frac{\partial q}{\partial P} \\ \frac{\partial p}{\partial Q} & \frac{\partial p}{\partial P} \end{bmatrix}$$

and an anti-symmetric matrix

$$A = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

According to Eq. 2 we also have that $cJAJ^T = A$ if J is the Jacobian of a symplectic map.

Writing out the calculation

$$JAJ^T = \begin{bmatrix} \frac{\partial q}{\partial Q} & \frac{\partial q}{\partial P} \\ \frac{\partial p}{\partial Q} & \frac{\partial p}{\partial P} \end{bmatrix} \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} \frac{\partial q}{\partial Q} & \frac{\partial p}{\partial Q} \\ \frac{\partial q}{\partial P} & \frac{\partial p}{\partial P} \end{bmatrix} =$$

$$= \begin{bmatrix} -\frac{\partial q}{\partial P} & \frac{\partial q}{\partial Q} \\ -\frac{\partial p}{\partial P} & \frac{\partial p}{\partial Q} \end{bmatrix} \begin{bmatrix} \frac{\partial q}{\partial Q} & \frac{\partial p}{\partial Q} \\ \frac{\partial q}{\partial P} & \frac{\partial p}{\partial P} \end{bmatrix} =$$

$$= \begin{bmatrix} -\frac{\partial q}{\partial Q}\frac{\partial q}{\partial P} + \frac{\partial q}{\partial Q}\frac{\partial q}{\partial P} & -\frac{\partial q}{\partial P}\frac{\partial p}{\partial Q} + \frac{\partial q}{\partial Q}\frac{\partial p}{\partial P} \\ -\frac{\partial p}{\partial P}\frac{\partial q}{\partial Q} + \frac{\partial p}{\partial Q}\frac{\partial q}{\partial P} & -\frac{\partial p}{\partial P}\frac{\partial p}{\partial Q} + \frac{\partial p}{\partial P}\frac{\partial p}{\partial Q} \end{bmatrix} =$$

$$= \begin{bmatrix} 0 & -\frac{\partial q}{\partial P}\frac{\partial p}{\partial Q} + \frac{\partial q}{\partial Q}\frac{\partial p}{\partial P} \\ -\frac{\partial q}{\partial Q}\frac{\partial p}{\partial P} + \frac{\partial q}{\partial P}\frac{\partial p}{\partial Q} & 0 \end{bmatrix} =$$

$$= \begin{bmatrix} 0 & -(\frac{\partial q}{\partial P}\frac{\partial p}{\partial Q} - \frac{\partial q}{\partial Q}\frac{\partial p}{\partial P}) \\ \frac{\partial q}{\partial P}\frac{\partial p}{\partial Q} - \frac{\partial q}{\partial Q}\frac{\partial Pp}{\partial P} & 0 \end{bmatrix} = \hat{A}$$

If the constant c in Eq. 2 is negative, the above turns into

$$JA^TJ^T = \begin{bmatrix} \frac{\partial q}{\partial Q} & \frac{\partial q}{\partial P} \\ \frac{\partial p}{\partial Q} & \frac{\partial p}{\partial P} \end{bmatrix} \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \frac{\partial q}{\partial Q} & \frac{\partial p}{\partial Q} \\ \frac{\partial q}{\partial P} & \frac{\partial p}{\partial P} \end{bmatrix} =$$

$$= \begin{bmatrix} \frac{\partial q}{\partial P} & -\frac{\partial q}{\partial Q} \\ \frac{\partial p}{\partial P} & -\frac{\partial p}{\partial Q} \end{bmatrix} \begin{bmatrix} \frac{\partial q}{\partial Q} & \frac{\partial p}{\partial Q} \\ \frac{\partial q}{\partial P} & \frac{\partial p}{\partial P} \end{bmatrix} =$$

$$= \begin{bmatrix} \frac{\partial q}{\partial Q}\frac{\partial q}{\partial P} - \frac{\partial q}{\partial Q}\frac{\partial q}{\partial P} & \frac{\partial q}{\partial P}\frac{\partial p}{\partial Q} - \frac{\partial q}{\partial Q}\frac{\partial p}{\partial P} \\ \frac{\partial p}{\partial P}\frac{\partial q}{\partial Q} - \frac{\partial p}{\partial Q}\frac{\partial q}{\partial P} & \frac{\partial p}{\partial P}\frac{\partial p}{\partial Q} - \frac{\partial p}{\partial P}\frac{\partial p}{\partial q} \end{bmatrix} =$$

$$= \begin{bmatrix} 0 & \frac{\partial q}{\partial P}\frac{\partial p}{\partial Q} - \frac{\partial q}{\partial Q}\frac{\partial p}{\partial P} \\ \frac{\partial q}{\partial Q}\frac{\partial p}{\partial P} - \frac{\partial q}{\partial P}\frac{\partial p}{\partial Q} & 0 \end{bmatrix} =$$

$$= \begin{bmatrix} 0 & \frac{\partial q}{\partial P}\frac{\partial p}{\partial Q} - \frac{\partial q}{\partial Q}\frac{\partial p}{\partial P} \\ -(\frac{\partial q}{\partial P}\frac{\partial p}{\partial Q} - \frac{\partial q}{\partial Q}\frac{\partial p}{\partial P}) & 0 \end{bmatrix} = \hat{A}^T$$

Setting

$$a = \frac{\partial q}{\partial P}\frac{\partial p}{\partial Q} - \frac{\partial q}{\partial Q}\frac{\partial p}{\partial P}$$

we get

$$\hat{A}\hat{A}^T = \begin{bmatrix} (-a)^2 & 0 \\ 0 & a^2 \end{bmatrix} = \begin{bmatrix} a^2 & 0 \\ 0 & a^2 \end{bmatrix}$$

If the original Jacobian J was that of a symplectic map then there should exist a constant c for which $ca^2 = 1$ everywhere.

A.4 Sym score calculation details

Polynomial regression To make the calculation of SyMetric computationally tractable we limit the maximal polynomial expansion threshold $\kappa = 5$. Furthermore, when evaluating the Jacobian in Eq. 3, we ignore the Jacobian terms with weight below $\epsilon = 1e-3$ to save computation. We also set the threshold for the goodness of fit parameter R^2 to $\alpha = 0.9$ to stop polynomial expansion and Lasso regression computation as soon as more than 90% of the variance in the ground truth trajectories is explained. We use the sklearn implementation of Lasso regression with maximum iteration number set to 1000, cross-validation set to 2, and Lasso $\alpha \in \{1e-8, 1e-7, 1e-6, 1e-5, 1e-4, 1e-3, 1e-2\}$. All trajectories are standardized before running the regression.

We find that in general the longer the trajectory on which the metric is calculated, the more accurate the results, however the accuracy on shorter trajectories (e.g. of 60 steps used in this paper) can be increased by calculating the metric on a number of samples and reporting the maximum or rounded average of the obtained score. In all our experiments we calculated the metric over 20 samples of 5 trajectories each. We found no improvements when we increased the number of trajectories within a sample.

When we use MLP instead of Lasso regression to measure SyMetric using the same amount of data, we find that the model overfits and the resulting symplecticity scores are very noisy $Sym = 0.38 \pm 0.18$, when the same model evaluated using the polynomial method gets $Sym = 0.0001 \pm 0.00005$, the latter score also being more indicative of the model's ability to extrapolate.

MLP To implement F as an MLP, we use 4 hidden layers with 4 units each, tanh nonlinearity, and Adam optimizer with $1.5e-3$ learning rate. To ensure that we do not overfit, we calculate the number of parameters in the MLP, and collect 1000x the parameter number of trajectories of 60 steps each. We then standardize the resulting data and split the resulting training set into randomly sampled mini-batches of size 64 for training the MLP, and train it for 10,000 steps. We also use L1 regularization on the weights with $\lambda = 0.01$. For calculating Sym, we use 50 training trajectories sampled at 10 random points each. To calculate SyMetric, we use $\alpha = 0.9$ and $\epsilon = 0.05$. See Alg. 2 for more details.

A.5 Understanding the constant

There are several ways to understand the need for the constant c in Eq. 3. First, since the learnt latent space is not grounded to the "ground truth", it can in effect learn to represent the state using different "units". Then the model will be representing exactly the same system in its latent space, but the energy values for all the trajectories will be orders of magnitude different from those of the ground truth trajectories, resulting in $E_m = cE_n$. Saying this, if all trajectories were related to each other with the same constant, then the mapping F could learn what the constant is, and one would expect $H_m(S) == H_n(F(S))$ to hold. This would be true if we were optimising for MSE when learning F, but since we optimise for variance explained (R^2), the magnitudes of $F(S)$ and s may be different, as long as they correlate. Second, some of the datasets used in the paper re-sample the constants of the Hamiltonian for each trajectory (e.g. mass in mass-spring +c). What this means is that the model is in effect learning multiple Hamiltonians from the same family. While in theory the model should be able to learn to infer the right constants from the pixel observations, as it does with the energy, so that $c = 1$, we choose not to penalise it for not doing so, and instead learning multiple Hamiltonians with different "units".

Algorithm 1: SyMetric algorithm using polynomial regression (PR).

Data: K ground truth state trajectories of T time steps $s=(q,p)\in\mathbb{R}^{K\times T\times 2n}$, K learnt latent state trajectories of T time steps $S=(Q,P)\in\mathbb{R}^{K\times T\times 2m}$, maximum polynomial expansion order κ, minimal acceptable goodness of fit α, maximum acceptable deviation for symplecticity ϵ

Result: SyMetric, Sym, R^2

```
begin
    p_exp ⟵ 1, ;
    R^2 ⟵ 0, ;
    Sym ⟵ ∅;
    A ∈ ℝ^{2n×2n} ⟵ [0  𝕀; −𝕀  0];
    S ⟵ RemoveUninformativeDims(S);
    while p_exp < κ and R^2 < α do
        S* ⟵ PolynomialExp(S, p_exp);
        F ⟵ Lasso(S*, s);
        R^2 ⟵ VarianceExplained(F(S*), s);
        p_exp ⟵ p_exp + 1 ;
    end
    J ⟵ ∂F/∂S
    for k ∈ K do
        𝕀̂ ⟵ ∅;
        for t ∈ T do
            J_kt ⟵ Evaluate(J, S*_kt);
            Add JAJ^T(JA^TJ^T)^T to 𝕀̂;
        end
        c ⟵ CalculateNormalisingConst(𝕀̂);
        Add MSE(c𝕀̂, 𝕀) to Sym;
    end
    Sym ⟵ Mean(Sym);
    if R^2 > α and Sym < ϵ then
        SyMetric ⟵ 1
    else
        SyMetric ⟵ 0
    end
end
```

Algorithm 2: SyMetric algorithm using MLP with ridge regularization.

Data: K ground truth state trajectories of T time steps $s=(q,p)\in\mathbb{R}^{K\times T\times 2n}$, K learnt latent state trajectories of T time steps $S=(Q,P)\in\mathbb{R}^{K\times T\times 2m}$, maximum polynomial expansion order κ, minimal acceptable goodness of fit α, maximum acceptable deviation for symplecticity ϵ

Result: SyMetric, Sym, R^2

```
begin
    p_exp ⟵ 1, ;
    R^2 ⟵ 0, ;
    Sym ⟵ ∅;
    A ∈ ℝ^{2n×2n} ⟵ [0  𝕀; −𝕀  0];
    S ⟵ RemoveUninformativeDims(S);
    F ⟵ MLP(S*, s);
    J ⟵ ∂F/∂S
    for k ∈ K do
        𝕀̂ ⟵ ∅;
        for t ∈ T do
            J_kt ⟵ Evaluate(J, S*_kt);
            Add JAJ^T(JA^TJ^T)^T to 𝕀̂;
        end
        c ⟵ CalculateNormalisingConst(𝕀̂);
        Add MSE(c𝕀̂, 𝕀) to Sym;
    end
    Sym ⟵ Mean(Sym);
    if R^2 > α and Sym < ϵ then
        SyMetric ⟵ 1
    else
        SyMetric ⟵ 0
    end
end
```