Imposing Conservation Properties in Deep Dynamics Modeling via Contrastive Learning Supplementary Material, ICLR 2023

A More Experiment Details

Table 2: Experimental details

	Ideal spring mass	Chemical kinematics	Kepler system	Heat equation
Dimension	2	2	4	101
Dynamics	$\dot{x}[1] = x[2]$ $\dot{x}[2] = -x[1]$	$\dot{x}[1] = -\kappa_1 x[1] + \kappa_2 x[2]$ $\dot{x}[2] = +\kappa_1 x[1] - \kappa_2 x[2]$	$\dot{x}[1] = x[3]$ $\dot{x}[2] = x[4]$ $r = \sqrt{x[1]^2 + x[2]^2}$ $\dot{x}[3] = -\frac{x[1]}{r^3}$ $\dot{x}[4] = -\frac{x[2]}{r^3}$	$\frac{\partial U}{\partial t} = \frac{\partial^2 U}{\partial y^2}$
Conservations	$x[1]^2 + x[2]^2 = C$	$x[1] + x[2] = C$	$\frac{x[3]^2+x[4]^2}{2} - \frac{1}{\sqrt{x[1]^2+x[2]^2}} = C_1$ $x[1]x[4] - x[2]x[3] = C_2$	$\int U dx = C$
Learning rate	1e-3	1e-3	1e-3	1e-3
Training traj. length (s)	3	7	25	10
# of points in training traj.	30	70	250	100
Eval. traj. length (s)	100	10	10	1
# of points in eval. traj.	1000	1000	1000	1000
# of traj.	50	50	200	200
Constrastive Learning Batch size	10	10	10	10
Constrastive Learning Epochs	1000	1000	1500	2000
Dynamics Learning Batch size	100	100	100	100
Dynamics Learning Epochs	1000	1000	1000	1000

All the dynamical modeling and contrastive learning neural networks mentioned in the paper are fully connected neural network, with 1 hidden layer of 100 neurons and tanh activations. The autoencoder used for the heat equation is also fully connected, both encoder and decoder use 2 hidden layers (32,16 neurons) and tanh activations. We conduct all the experiments on a single 2080Ti GPU.

Table 3: Simulation error (**log scale**) over the tasks

Task	Mean square error		Violation of conservation laws	
	Baseline NN	ConCerNet	Baseline NN	ConCerNet
Ideal spring mass system	-0.823 $\pm$ 0.479	**-1.126 $\pm$ 0.414**	-1.250 $\pm$ 0.795	**-3.213 $\pm$ 1.041**
Chemical kinematics	-1.182 $\pm$ 0.039	**-1.924 $\pm$ 0.967**	-1.626$\pm$ 0.200	**-2.584 $\pm$ 0.227**
Kepler system	-0.071 $\pm$ 0.055	**-0.485 $\pm$ 0.035**	-1.224 $\pm$ 0.075	**-2.297 $\pm$ 0.599**
Heat equation	-0.905 $\pm$ 0.210	**-1.090 $\pm$ 0.316**	0.255 $\pm$ 0.340	**-0.759 $\pm$ 0.111**

Table 3 shows the same result of Table 1 in log scale. As the system error exponentially grows with time during the prediction simulation, results under log scale better illustrate the consistent performance improvement with our method.

Table 4: Simulation error comparison with previous work (HNN (Greydanus et al., 2019))

Task	Mean square error			Violation of conservation laws		
	Baseline NN	ConCerNet	HNN	Baseline NN	ConCerNet	HNN
Ideal spring mass	0.209 $\pm$ 0.172	**0.076 $\pm$ 0.063**	0.110$\pm$0.099	0.096 $\pm$ 0.080	**2.1e-3 $\pm$ 1.9e-3**	2.7e-3 $\pm$ 3.4e-3
Kepler system	0.854 $\pm$ 0.103	0.328 $\pm$ 0.026	**0.194 $\pm$ 0.085**	0.060 $\pm$ 0.011	9.5e-3 $\pm$ 1.22e-2	**8.7e-3 $\pm$ 1.41e-2**

Table 4 shows the results comparison between our proposed method and HNN (Greydanus et al., 2019) on the two examples. In general, the two methods show similar performance (each method wins one experiment), and their conservation and coordinate errors are much smaller than the vanilla NN. HNN is not appliable to the other two experiments as they are not Hamiltonian systems.

Table 5: Hyper-parameter study in heat equation experiment

Dimensions	$\dim(z) = 8$		$\dim(z) = 9$		$\dim(z) = 10$	
	MSE	Vio. of Cons.	MSE	Vio. of Cons.	MSE	Vio. of Cons.
$\dim(H) = 1$	0.091 ± 0.081	$\mathbf{0.051 \pm 0.014}$	0.098 ± 0.073	0.178 ± 0.045	0.096 ± 0.046	0.107 ± 0.038
$\dim(H) = 2$	0.105 ± 0.062	0.082 ± 0.028	$\mathbf{0.063 \pm 0.018}$	0.071 ± 0.014	0.070 ± 0.057	0.065 ± 0.012
$\dim(H) = 3$	0.607 ± 0.328	0.467 ± 0.234	0.505 ± 0.369	0.597 ± 0.329	0.210 ± 0.146	0.163 ± 0.061

Table 5 shows the hyper-parameter study for the heat equation problem. We play with three autoencoder latent space dimensions ($\dim(z) = 8, 9, 10$) and three conservation space dimensions ($\dim(H) = 1, 9, 10$). We found the model performance is less affected by the autoencoder space dimension. However, both coordinate and conservation errors grow significantly when using 3 as the conservation function output dimension. This can be explained by the over-parameterization of the conservation space. If we use more conservation dimensions than the actual number of conservation laws, the extra dimension constraint limits the flexibility of the dynamical model representation power. When $\dim(H(x)) = 3$, we also notice the performance decreasing trend when using a smaller latent space dimension, this indicates the dynamical prediction model is under heavy constraint and needs extra space to learn the data.

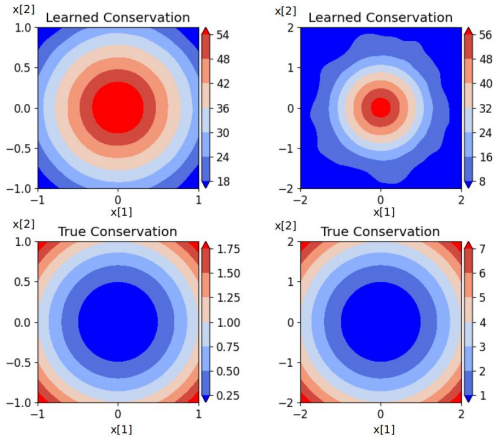

Figure 7: Learned spring-mass system conservation function vs ground truth, left: on $[-1, 1] \times [-1, 1]$, right: on $[-2, 2] \times [-2, 2]$

Figure 7 shows the conservation learning model performance on out-of-distribution data for the spring-mass system. During training, the norm of the system state ($\sqrt{x[1}^2 + x[2]^2$)is randomly sampled from $[0.3, 1.2]$. Therefore, the model shows relatively good performance for states within the unit circle and cannot capture the square function for states with a norm greater than 1.5. Similar to many data-driven methods, this is expected as the model has never been exposed to the testing distribution before.

B Encourage "correct" sign of the learned invariant function to the exact conservation law

For a conserved system, the observation lacks the directional information of conservation function, making it impossible for the neural network to figure out the correct sign. However, such information can be extracted from a dissipative system. For a slightly "leaky" system, we propose the following

ranking loss function to encourage the right order in learned invariants by utilizing the directional information:

$$\mathcal{L}_{\text{rank}} = \frac{2}{NT(T-1)} \sum_{i=1}^{N} \sum_{t_1=1}^{T} \sum_{t_2=t_1+1}^{T} \phi(H_\theta(x^i_{t_2}) - H_\theta(x^i_{t_1})) \tag{12}$$

where ϕ is an elementwise clip function $\phi(x) = \text{clip}(1 + x, 1, 2)$ and the loss function penalizes the model if the learned quantity of later time steps are great than earlier steps. To bypass the tuning of hyperparameter, we simply use the overall loss augmented by multiplication instead of summation: $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{rank}} * \mathcal{L}_{\text{con}}$, then practice the loss function on a **Real spring mass system** with friction terms

$$\dot{x}[1] = x[2]$$
$$\dot{x}[2] = -x[1] - 0.02x[2]$$

and fit the learned function to the exact conservation law by an affine function. We conducted the experiments over 10 different seeds and record the coefficients of the optimal fitting function

$$\beta_{0,opt}, \beta_{1,opt} = \arg\min_{\beta_0,\beta_1} \mathbb{E}_{x\sim C\sim D} \left[\|\beta_1 H_{\theta_c}(x) + \beta_0 - \text{consv}(x)\|^2 \right] \tag{13}$$

With the ranking loss, the neural network is capable to "recognize" the correct sign of the conservation function with minor influence on fitting error.

Table 6: Relative coefficient between learned invariants and exact conservation function

	# of positive $\beta_{1,opt}$	$\beta_{1,opt}$	$\beta_{0,opt}$	fitting error
w/o ranking loss	6/10	6.95± 33.11	-6.31± 33.21	0.283 ± 0.057
w/ ranking loss	10/10	31.76± 3.52	-22.14 ± 5.70	0.231± 0.052

C ADDITIONAL NUMERICAL EXAMPLES

C.1 QUADRATIC $g(x)$

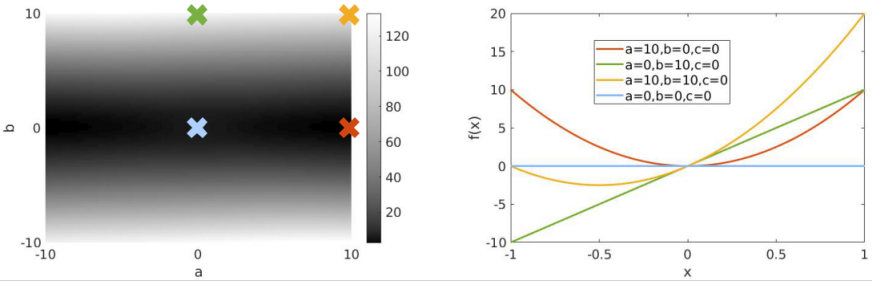

Figure 8: Right: $h_\theta(x) = ax^2 + bx + c$ on different parameterization. Left: Contrastive loss on 1D continuous label space as function of $(a, b) \in [-10, 10]^2, \epsilon = 0.05$.

We consider another numerical experiment with scalar function $g(x) = x^2$ and all other setup being same as the example in the main paper (Section 5.2). We found the optimal solutions over $(a, b) \in [-10, 10]^2$ are $h(x) = 10x^2/-10x^2$, when $h_\theta(x)$ is a linear function to $g(x)$.

C.2 $g(x)$ WITH TWO PEAKS

To show the model capability for learning more complex conservation function terrains (i.e. energy functions with multiple peaks), we design $g(x) = -\cos(2\pi x)$ with two different peaks at $x = -0.5$ and $x = 0.5$. We use a 4th order polynomial ($h_\theta(x) = ax^4 + bx^2 + c$) to parameterize

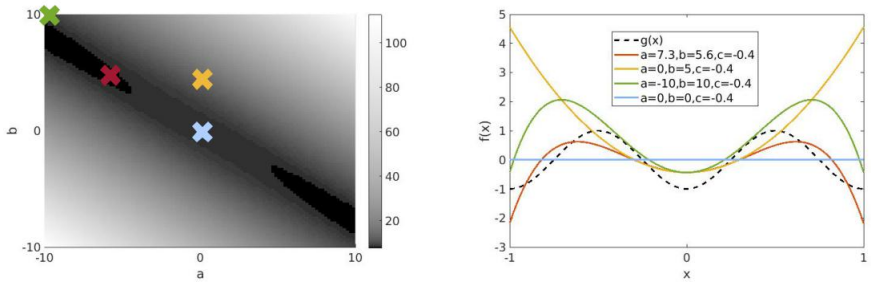

Figure 9: Right: $h_\theta(x) = ax^4 + bx^2 + c$ on different parameterization. Left: Contrastive loss on 1D continuous label space as function of $(a, b) \in [-10, 10]^2, \epsilon = 0.05$.

the conservation function. The optimal solution lies at $a = 7.3$ and $b = -5.6$ in our numerical experiment. This is close to the 4th order polynomial fitting on $g(x) = -\cos(2\pi x)$ on $x \in [-1, 1]$: $-7.3004x^4 + 5.6915x^2 - 0.4366$.

D Contrastive Learning Hyperparameter Selection

To evaluate how "good" the learned model is, we introduce two metrics here and conduct experiments on the **Ideal spring mass system**. First, we use the fitting error from Equation (13), revealing how well the model approximate the exact physics law under linear fitting.

$$\text{fitting error} = \min_{\beta_0, \beta_1} \mathbb{E}_{x \sim C \sim D} \left[\|\beta_1 H_{\theta_c}(x) + \beta_0 - \text{consv}(x)\|^2 \right] \tag{14}$$

The second metric is the contrastive classification accuracy, where a state point $x^i_{t_1}$ is correctly classified if the probability drawn from Equation (3) is greater than 0.5, s.t.

$$\frac{\sum_{t_2=1}^{T} \exp(-\|H_{\theta_c}(x^i_{t_1}) - H_{\theta_c}(x^i_{t_2})\|^2) \mathbb{1}(t_1 \neq t_2)}{\sum_{j=1}^{N} \sum_{t_2=1}^{T} \exp(-\|H_{\theta_c}(x^i_{t_1}) - H_{\theta_c}(x^j_{t_2})\|^2) \mathbb{1}(i \neq j \text{ or } t_1 \neq t_2)} > 0.5 \tag{15}$$

Table 7: Averaged learned invariant fitting error to exact conservation and classification accuracy (both $\times 0.01$)

# of traj. / Batch size	10	25	50	200	400
	err./acc.	err./acc.	err./acc.	err./acc.	err./acc.
5	8.2/81.3	7.4/81.6	4.9/85.3	2.4/88.3	1.7/89.1
10	10.2/60.1	8.1/72.0	5.6/70.7	2.7/75.4	1.9/72.4
20	N/A	8.6/68.3	6.1/59.3	3.0/46.4	2.2/45.3
50	N/A	N/A	6.8/8.3	3.2/9.1	2.3/9.2

We perform a parametric study for 4 different training batch size and number of trajectories used and record the averaged results over 5 random seeds in Table 7. Each training lasts 1000 epochs, which is long enough for loss function to stabilize. As expected, the classification accuracy increases with more data. Interestingly, we found the fitting error decrease proportionally to the inverse of the square root of the trajectory number, s.t. err $\sim \mathcal{O}(N^{-1/2})$. We also notice the performance drop with larger batch size. For classification accuracy, the drop is significant with Equation (15) heavily biased on negative examples. As for the increased error with large batch size, we suspect it is because large mini batches are more likely to involve "similar" trajectories, making it more difficult to distinguish the correct one as other close trajectory might have similar conservation values.