Physics-Augmented Learning: A New Paradigm Beyond Physics-Informed Learning

Ziming Liu
MIT & IAIFI
zmliu@mit.edu

Yunyue Chen
KCL
yunyue.chen@kcl.ac.uk

Yuanqi Du
GMU
ydu6@gmu.edu

Max Tegmark
MIT & IAIFI
tegmark@mit.edu

Abstract

Integrating physical inductive biases into machine learning can improve model generalizability. We generalize the successful paradigm of physics-informed learning (PIL) into a more general framework that also includes what we term physics-augmented learning (PAL). PIL and PAL complement each other by handling *discriminative* and *generative* properties, respectively. In numerical experiments, we show that PAL performs well on examples where PIL is inapplicable or inefficient.

1 Introduction

Physics-informed learning (PIL) [1, 2, 3, 4, 5] has been widely used in scientific applications where physical inductive biases are applicable. The integration of domain knowledge into machine learning can not only enhance generalization, but also make models more interpretable. However, PIL implicitly requires physics properties to be *discriminative*, as opposed to *generative* (defined below). To complement PIL, we propose a new paradigm call physics-augmented learning (PAL) to handle generative properties, as illustrated in Figure 1. We define and compare *discriminative* and *generative* properties in Section 2, propose PAL in Section 3 and compare it with PIL, and demonstrate PAL's effectiveness via numerical experiments in Section 4.

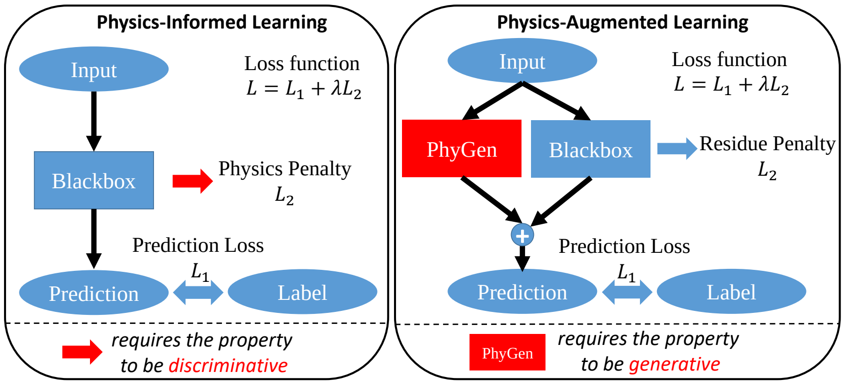


Figure 1: Compare Physics-informed learning (PIL, left) and physics-augmented learning (PAL, right). PIL and PAL apply to discriminative and generative properties respectively.

2 Discriminative and Generative Properties

What is a property? A property P is a mapping from an object f to a boolean variable: $P(f)$ is true if f satisfies P, false otherwise. For example, if f refers to an individual with age a, and P is the statement that "The individual has the age no more than 30.", then $P(f)$ is true if $a \leq 30$ and $P(f)$ is false if $a > 30$.

Inspired by Generative adversarial networks (GAN) [6], we define a *generator* and a *discriminator* associated with a property P.

Definition 2.1 (Generator). *A generator can generate objects f that have the property P.*

Definition 2.2 (Discriminator). *A discriminator determines if an input object f has the property P or not.*

To be maximally useful, a generator should be implementable as a symbolic formula or feedforward neural network, and a discriminator can be implementable as a classifier or a loss function. More specifically, we define *ideal* generators and discriminators as follows:

Definition 2.3 (Ideal Generator). *An ideal generator is (1) accurate (never generate an f such that $P(f)$ is false), (2) complete (can generate any f such that $P(f)$ is true), (3) efficient (can generate f in polynomial time), and (4) differentiable (can exploit derivative-based optimization methods such as back propagation).*

Definition 2.4 (Ideal Discriminator). *An ideal discriminator is (1) accurate (always computes $P(f)$ correctly), (2) efficient (computes $P(f)$ in polynomial time), and (3) differentiable (can exploit derivative-based optimization methods such as back propagation). In this paper, we deal with one specific ideal discriminator $\hat{L}$ such that $\hat{L}f = 0$ when $P(f)$ is true and $\hat{L}f \neq 0$ when $P(f)$ is false.*

We now define discriminative and generative properties:

Definition 2.5 (Generative property). *A property P is generative if there exists an ideal generator for P.*

Definition 2.6 (Discriminative property). *A property P is discriminative if there exists an ideal discriminator for P.*

Let us clarify these abstract definitions with a few examples of properties below, summarized in Table 1).

A. Lagrangian property For a physics system with generalized coordinate $\mathbf{q}$ and velocity $\dot{\mathbf{q}}$, the acceleration field $\ddot{\mathbf{q}}$ is *Lagrangian* if there exists a Lagrangian function $\mathcal{L}(\mathbf{q}, \dot{\mathbf{q}})$ such that $\ddot{\mathbf{q}} = (\nabla_{\dot{\mathbf{q}}}\nabla_{\dot{\mathbf{q}}}^T\mathcal{L})^{-1}(\nabla_{\mathbf{q}}\mathcal{L} - (\nabla_{\mathbf{q}}\nabla_{\dot{\mathbf{q}}}^T\mathcal{L})\dot{\mathbf{q}})$ [7, 8]. The Lagrangian property is generative by definition but not discriminative, because (perhaps surprisingly) there is no known efficient method to determine whether a given $\ddot{\mathbf{q}}$ is Lagrangian or not [8].

B. Positive definiteness By analogy with linear operators, we say that a function $f(x)$ is *positive definite* if there exists a function g such that $f(x) = g(g(x))$. For example, the function that time-evolves a physical system during an interval ΔT is positive definite, since it is equivalent by time-evolving by $\Delta T/2$ twice. The positive definiteness property is generative by definition, but not discriminative to the best of our knowledge.

C. Manifest symmetry Many manifest symmetries are discriminative, with associated discriminators that can be elegantly described by partial differential equations [9]. For example, a vector field $\mathbf{f}(\mathbf{x})$ is symmetric under a Lie group $\mathcal{G}$ if $\mathbf{f}(g\mathbf{x}) = g(\mathbf{f}(\mathbf{x}))$ for all $g \in \mathcal{G}$. This symmetry property is discriminative because it is equivalent to $(K_i\mathbf{x} \cdot \nabla - K_i)\mathbf{f} = 0$ for all group generators K_i[9]. It is also generative due to recent advances in equivariant neural networks [10, 11, 12, 13, 14].

D. Hidden symmetry However, hidden symmetries are not discriminative, because they require coordinate transformations to 'generate' manifestly symmetric objects [9]. For example, manifest Hamiltonicity is shown to be discriminative in [9], but hidden Hamiltonicity is a generative property and that is equivalent to Lagrangianess [8].

Table 1: Generative and discriminative properties

Properties	A. Lagrangian Property	B. Positive Definiteness	C. Manifest Symmetry	D. Hidden Symmetry	E. Separability	F. PDE Satisfiability
Generative	Yes	Yes	Yes	Yes	Yes	Yes
Discriminative	No	No	Yes	No	Yes	Yes

Table 2: Neural network and loss function for PIL and PAL on the separability example

Paradigm	NN	Prediction error L_1, Penalty L_2
PIL	$f(x_1, x_2; \theta)$	$\frac{1}{N}\sum_i \lvert f_0(x_1^{(i)}, x_2^{(i)}) - f(x_1^{(i)}, x_2^{(i)}; \theta)\rvert$, $\frac{2}{N(N-1)}\sum_{j>i} \lvert f(x_1^{(i)}, x_2^{(i)}; \theta) + f(x_1^{(j)}, x_2^{(j)}; \theta) - f(x_1^{(i)}, x_2^{(j)}; \theta) - f(x_1^{(j)}, x_2^{(i)}; \theta)\rvert$
PAL	$f_1(x_1; \theta_1), f_2(x_2; \theta_2)$, $f_{12}(x_1, x_2; \theta_{12})$	$\frac{1}{N}\sum_i \lvert f_0(x_1^{(i)}, x_2^{(i)}) - (f_1(x_1^{(i)}; \theta_1) + f_2(x_2^{(i)}; \theta_2)) - f_{12}(x_1^{(i)}, x_2^{(i)}; \theta_{12})\rvert$, $\frac{1}{N}\sum_i \lvert f_{12}(x_1^{(i)}, x_2^{(i)}; \theta_{12})\rvert$

E. Separability: A differentiable bivariate function $f(x_1, x_2)$ is (additively) separable if there exists two unary functions f_1 and f_2 such that $f(x_1, x_2) = f_1(x_1) + f_2(x_2)$. Separability is generative by definition, and also discriminative because it is equivalent to $\hat{L}f \equiv \partial^2 f/\partial x_1 \partial x_2 = 0$ [15, 16].

F. PDE satisfiability We say a function $f(x, t)$ satisfies a partial differential equation (PDE) if $g(t, f, f_t, f_x, ..) = 0$. This property is discriminative by definition, with $\hat{L}f = g$. It is also generative when $f(x, t)$ can be efficiently computed by a numerical PDE solver given proper boundary conditions; this idea underlies neural ordinary/partial/stochastic differential equations [17, 18, 19].

3 Physics-informed Learning (PIL) and Physics-augmented learning (PAL)

In this section, we first review physics-informed learning (PIL) and its limitations, motivating our proposed physics-augmented learning (PAL) framework.

3.1 Physics-informed Learning (PIL)

The essence of PIL is to seamlessly integrate data and mathematical physics models, and a common way is to add a soft penalty term L_2 (corresponding to physics properties) to the prediction loss L_1 [1] (Figure 1, left panel). PIL works for *discriminative* properties. Indeed, one of its greatest successes lies in solving forward/inverse PDE problems [2, 3, 5], based on the unstated fact that satisfying a PDE is discriminative. We clarify PIL with a toy example below.

Example: PIL for separability Suppose that our training dataset (N samples) is generated by the oracle $y = f_0(x_1, x_2)$ where $f_0 : \mathbb{R}^2 \to \mathbb{R}$ and that we want to use a parametrized neural network $f(x_1, x_2; \theta)$ to fit the data with a function $f(x_1, x_2; \theta)$ that is additively separable. PIL does this using a loss function with two terms: $L \equiv L_1 + \lambda L_2$, where the prediction loss L_1 and separability loss L_2 are

$$L_1(\theta) \equiv \frac{1}{N}\sum_i \lvert f_0(x_1^{(i)}, x_2^{(i)}) - f(x_1^{(i)}, x_2^{(i)}; \theta)\rvert, \quad L_2(\theta) \equiv \frac{1}{N}\sum_i \left\lvert \frac{\partial^2 f(x_1^{(i)}, x_2^{(i)}; \theta)}{\partial x_1 \partial x_2} \right\rvert,$$

and the constant $\lambda > 0$ is a penalty coefficient.[1]

By definition, each discriminative property P can be written $\hat{L}f = 0$ for some operator $\hat{L}$, so $L_2 \equiv \lvert \hat{L}f \rvert$ is a natural measure of property violation. In contrast, non-discriminative properties, *e.g.*, being Lagrangian or positive definite, lack a known efficiently computable criterion $\hat{L}f = 0$. Fortunately, the PAL framework proposed below can come to rescue whenever the properties of interest are *generative*.

[1] Alternatively, L_2 can be expressed in terms of finite differences instead of derivatives; we do this for our numerical experiments. For example, additive separability corresponds to the condition listed in Table 2, which we average over all pairs of data points.

3.2 Physics-augmented Learning (PAL)

Although both PIL and PAL aim to leverage inductive biases in machine learning, their approaches are quite different: while PIL is based on regularization design, PAL is based on model design. Useful regularization designs and model designs are only available for properties that are discriminative and generative, respectively.

In the PAL paradigm, the whole model consists of two parallel modules (see Figure 1, right panel): the first module (`PhyGen`) strictly satisfies the generative property, and the second module (`Blackbox`) *augments* the expressive power to allow violation of the property. The loss function consists of two terms: the standard prediction error L_1, and a penalty term L_2 defined as some norm of `Blackbox` module output. The combined loss function is $L = L_1 + \lambda L_2$.

Example: PAL for separability In PAL we have two neural modules `PhyGen` and `Blackbox`. `PhyGen` satisfies strictly the additive separability by having two sub-networks $(f_1(x_1;\theta_1), f_2(x_2;\theta_2))$ and outputs their sum. `Blackbox` is a fully-connected neural network $f_{12}(x_1, x_2; \theta_{12})$ that can universally approximate any two-variable continuous function. The whole prediction is thus $(f_1(x_1;\theta_1) + f_2(x_2;\theta_2)) + f_{12}(x_1, x_2; \theta_{12})$ and the prediction loss L_1 is its distance from the label $y = f_0(x_1, x_2)$. The penalty loss is simply the function norm of the `Blackbox` *i.e.*, $L_2 = |f_{12}(x_1, x_2; \theta_{12})|$. Our PIL and PAL examples are summarized and compared in Table 2.

When should I use PAL rather than PIL? PIL and PAL are complementary frameworks exploiting discriminative and generative properties, respectively. As a consequence, one should resort to PAL without hesitation when the property of interest is generative and non-discriminative. Another reason to use PAL is that it provides a model decomposition into `PhyGen` and `Blackbox`, and interpreting them potentially produces physical insights. For example, the decomposition into conservative and non-conservative force fields enabled new-physics discovery in [8].

How to choose the hyperparameter λ? It was recently proven that $L = L_1 + \lambda L_2$ produces a phase transition at $\lambda = 1$ [8][2]. $\lambda > 1$ is the undesirable phase, so in principle one can simply choose any $\lambda < 1$ to obtain vanishing prediction loss; the numerical results of [8] suggest that $\lambda \in [0.02, 0.5]$ produces accurate and robust results in practice. We verify these observations with the additive separability example in Appendix B. In our numerical experiments, we choose $\lambda = 0.2$ if not stated otherwise.

4 Numerical Experiments

We demonstrate the effectiveness of PAL on two tasks: symbolic regression and dynamics prediction. PAL performs well on these applications, while PIL is inapplicable or performs worse than PAL.

4.1 Symbolic Regression

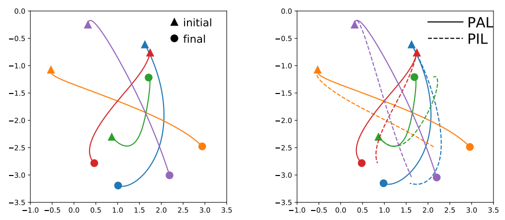
The goal of symbolic regression is to find a symbolic expression that matches data from an unknown function f. The physics-inspired AI Feynman symbolic regression module [15, 16] tests if a dataset satisfies certain properties, including symmetries. However, AI Feynman can only discovers and exploits these if they hold with high accuracy. To relax this, we employ PAL to first decompose the function f into two parts: a property-satisfying part f_+ and a property-violating residual f_-. We then apply AI Feynman to both parts separately to obtain symbolic formulas; f_+ satisfies the strict property which AI Feynman can exploit.

We experiment with three properties: additive separability, rotational invariance and positive definiteness. Training details can be found in Appendix B. Table 3 shows the symbolic regression results: for the first two properties, PAL decomposes the function as desired while PIL can only learn the whole function; for the positivity example, PIL is not applicable while PAL can extract meaningful partial function g.

[2]Here the loss functions L_1 and L_2 should be defined as norms to produce sharp phase transition behavior, *e.g.*, mean-absolute error (MAE) or Euclidean norm. In contrast, MSE does not produce a sharp phase transition.

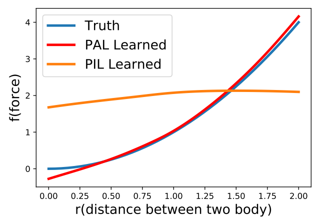
Table 3: Symbolic regression results

Property	Methods	f_+	f_-
Additive Separability $f = x_1^2 + x_2^2 + x_1 x_2$	Truth	$(x_1^2) + (x_2^2)$	$x_1 x_2$
	AI Feynman + PAL	$(x_1^2 - 0.02) + (x_2^2 - 0.01)$	$x_1 x_2 + 0.03$
	AI Feynman + PIL	$x_1^2 + x_2^2 + x_1 x_2$	
Rotational Invariance $f = 0.5(x_1^2 + x_2^2) + 0.32x_1$	Truth	$0.5(x_1^2 + x_2^2)$	$0.32x_1$
	AI Feynman + PAL	$0.5(x_1^2 + x_2^2)$	$0.31998x_1$
	AI Feynman + PIL	$0.5(x_1^2 + x_2^2) + 0.32x_1$	
Positivity $f = \sin(\sin(x)) + 0$	Truth	$\sin(\sin(x))$	0
	AI Feynman + PAL	$g(g(x)), g = -\sin(x) + 0.004$	0
	AI Feynman + PIL	Not Applicable	



Ground Truth PAL & PIL

(a) Trajectory



(b) Force

Figure 2: Dynamics prediction results: (a) Trajectory; (b) Force.

4.2 Dynamics prediction: N-body Problem

In this example, we aim to learn N-body dynamics from the initial $(t = 0)$ and final $(t = 1)$ states of N particles. We assume that these unit mass particles $(m = 1)$ obey the Newtonian mechanics with pairwise interactions, *i.e.*, particle i exerts a force $\mathbf{f}_{ij} = f(|\mathbf{x}_i - \mathbf{x}_j|)\frac{\mathbf{x}_i - \mathbf{x}_j}{|\mathbf{x}_i - \mathbf{x}_j|_2}$ on particle j. Our ground truth is that $f(r) = r^2$ when computing final states from initial states, but we pretend not to know f and aim to infer it solely from the initial and final states (indicated by triangles and dots in Figure 2). The property P that we explore with PAL and PIL is the time-independence of f, which is clearly both discriminative and generative. Details of loss functions, data generation and training details are included in Appendix A and C. Figure 2 shows that PAL outperforms PIL in terms of both trajectory interpolation and force recovery. In other words, although both PIL and PAL can be applied for this example, PAL reveals the underlying dynamics much more accurately that PIL.

5 Conclusions & Discussions

We have proposed a new paradigm called physics-augmented learning (PAL) to effectively integrate physical properties into unconstrained neural networks. PAL complements the well-known physics-informed learning (PIL) paradigm, by applying in some cases where PIL is inapplicable and by outperforming PIL in some cases where both can be used. While PIL is based on regularization design and applies to discriminative properties, PAL is based on model design and applies to generative properties.

Although PAL in its general form is explicitly formulated for the first time in this paper, examples of it have been implicitly adopted in many successful machine learning models owing to its ability to integrate human knowledge into the model design phase. For example, AlphaFold2 demonstrated that designing deep learning models with proper inductive biases could give superior performance for an unsolved grand challenge [20]. There is growing interest in the ML community in how inductive biases can shape deep learning models; for example, [21] discovers how two fundamental deep learning models (CNNs and Transformers) leverage their own inductive biases and tackle challenges in the computer vision domain. We are hopeful that the PAL-PIL framework unifying inductive biases can lead to further advances in the ML community for tackling real-world challenges with optimal selection and design of inductive biases.

Acknowledgement

We would like to thank Silviu Udrescu and Jiaxi Zhao for valuable discussions. We thank the Center for Brains, Minds, and Machines (CBMM) for hospitality. This work was supported by The Casey and Family Foundation, the Foundational Questions Institute, the Rothberg Family Fund for Cognitive Science and IAIFI through NSF grant PHY-2019786.

References

[1] George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

[2] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations, 2017.

[3] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics informed deep learning (part ii): Data-driven discovery of nonlinear partial differential equations, 2017.

[4] M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[5] Maziar Raissi, Alireza Yazdani, and George Em Karniadakis. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science*, 367(6481):1026–1030, 2020.

[6] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[7] Miles Cranmer, Sam Greydanus, Stephan Hoyer, Peter Battaglia, David Spergel, and Shirley Ho. Lagrangian neural networks. *arXiv preprint arXiv:2003.04630*, 2020.

[8] Ziming Liu, Bohan Wang, Qi Meng, Wei Chen, Max Tegmark, and Tie-Yan Liu. Machine-learning non-conservative dynamics for new-physics detection, 2021.

[9] Ziming Liu and Max Tegmark. Machine-learning hidden symmetries, 2021.

[10] Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International conference on machine learning*, pages 2990–2999. PMLR, 2016.

[11] Nathaniel Thomas, Tess Smidt, Steven Kearnes, Lusann Yang, Li Li, Kai Kohlhoff, and Patrick Riley. Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds. *arXiv preprint arXiv:1802.08219*, 2018.

[12] Fabian B Fuchs, Daniel E Worrall, Volker Fischer, and Max Welling. Se (3)-transformers: 3d roto-translation equivariant attention networks. *arXiv preprint arXiv:2006.10503*, 2020.

[13] Risi Kondor, Zhen Lin, and Shubhendu Trivedi. Clebsch-gordan nets: a fully fourier space spherical convolutional neural network. *arXiv preprint arXiv:1806.09231*, 2018.

[14] Victor Garcia Satorras, Emiel Hoogeboom, and Max Welling. E (n) equivariant graph neural networks. *arXiv preprint arXiv:2102.09844*, 2021.

[15] Silviu-Marian Udrescu and Max Tegmark. Ai feynman: A physics-inspired method for symbolic regression. *Science Advances*, 6(16), 2020.

[16] Silviu-Marian Udrescu, Andrew Tan, Jiahai Feng, Orisvaldo Neto, Tailin Wu, and Max Tegmark. Ai feynman 2.0: Pareto-optimal symbolic regression exploiting graph modularity. *arXiv preprint arXiv:2006.10782*, 2020.

[17] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations, 2019.

[18] Jun-Ting Hsieh, Shengjia Zhao, Stephan Eismann, Lucia Mirabella, and Stefano Ermon. Learning neural pde solvers with convergence guarantees, 2019.

[19] Patrick Kidger, James Foster, Xuechen Li, Harald Oberhauser, and Terry Lyons. Neural sdes as infinite-dimensional gans, 2021.

[20] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, pages 1–11, 2021.

[21] Maithra Raghu, Thomas Unterthiner, Simon Kornblith, Chiyuan Zhang, and Alexey Dosovitskiy. Do vision transformers see like convolutional neural networks? *arXiv preprint arXiv:2108.08810*, 2021.

Supplementary material

A Definitions of PIL and PAL losses

In Section 4, we conducted experiments related to four properties: additive separability, rotational invariance, positivity and time independence. PAL applies to all of these properties, while PIL is not applicable to positivity but applies to other three properties. In Table 4 we show the design of neural networks and loss functions for each property in the framework of PIL and PAL.

All neural networks $f_*(*;*)$ (each $*$ can be anything) are fully-connected networks which can approximate any continuous function of input variables (if wide enough). Their differences lie in input variables. For PIL, the only network is fully-connected and takes in all available variables. For PAL, we have a `PhyGen` and `Blackbox` which usually have different inputs. `Blackbox` takes in all available input variables while `PhyGen` usually takes in fewer variables which is determined by the specific property, as detailed below.

- Additive separability. `PhyGen`: f_1 takes in x_1 only, while f_2 takes in x_2 only. The output is $f_1(x_1;\theta_1) + f_2(x_2;\theta_2)$. `Blackbox`: f_{12} takes in both x_1 and x_2. The output is $f_{12}(x_1, x_2;\theta_{12})$.
- Rotational invariance. `PhyGen`: f_1 takes in one variable $R \equiv \sqrt{x_1^2 + x_2^2}$ only. The output is $f_1(R;\theta_1)$. `Blackbox`: f_2 takes in both x_1 and x_2. The output is $f_2(x_1, x_2;\theta_2)$.
- Positivity. `PhyGen`: f_1 takes in x and outputs $f_1(f_1(x;\theta_1);\theta_1)$. `Blackbox`: f_2 takes in x and outputs $f_2(x;\theta_2)$.
- Time independence. `PhyGen`: f_1 takes in r where r is the distance between two particles. The output is $f_1(r;\theta_1)$. `Blackbox`: f_2 takes in both r and t. The output is $f_2(r, t;\theta_2)$.

Table 4: PIL and PAL neural networks and loss functions for four properties

Property	Variables	Paradigm	NN	Prediction error L_1, Penalty L_2
Additive Separability	x_1, x_2	PIL	$f(x_1, x_2;\theta)$	$\frac{1}{N}\sum_i \lvert f_0(x_1, x_2) - f(x_1, x_2;\theta)\rvert$, $\frac{2}{N(N-1)}\sum_{j>i} \lvert f(x_1^{(i)}, x_2^{(i)};\theta) + f(x_1^{(j)}, x_2^{(j)};\theta) - f(x_1^{(i)}, x_2^{(j)};\theta) - f(x_1^{(j)}, x_2^{(i)};\theta)\rvert$
		PAL	$f_1(x_1;\theta_1), f_2(x_2;\theta_2)$, $f_{12}(x_1, x_2;\theta_{12})$	$\lvert f_0(x_1, x_2) - (f_1(x_1;\theta_1) + f_2(x_2;\theta_2)) - f_{12}(x_1, x_2;\theta_{12})\rvert$, $\lvert f_{12}(x_1, x_2;\theta_{12})\rvert$
Rotational Invariance	x_1, x_2	PIL	$f(x_1, x_2;\theta)$	$\lvert f_0(x_1, x_2) - f(x_1, x_2;\theta)\rvert$, $\lvert f(x_1, x_2;\theta) - f(x_1', x_2';\theta)\rvert\ [(x_1', x_2')^T = \text{Rotation}(\alpha)(x_1, x_2)^T]$
		PAL	$f_1(R;\theta_1)$, $f_2(x_1, x_2;\theta_2)$	$\lvert f_0(x_1, x_2) - f_1(R;\theta_1) - f_2(x_1, x_2;\theta_2)\rvert$, $\lvert f_2(x_1, x_2;\theta)\rvert$
Positivity	x	PIL	Not Applicable	Not Applicable
		PAL	$f_1(x;\theta_1)$, $f_2(x;\theta_2)$	$\lvert f_0(x) - f_1(f_1(x;\theta_1);\theta_1) - f_2(x;\theta_2)\rvert$, $\lvert f_2(x;\theta_2)\rvert$
Time Independence	r, t	PIL	$f(r, t;\theta)$	$\lvert \mathbf{z}(t=0) + \int_0^1 dt\mathbf{F}(\mathbf{z}, t) - \mathbf{z}(t=1)\rvert$, $\sum_{i=1}^n \lvert f(r, i\Delta t;\theta) - f(r, (i+1)\Delta;\theta)\rvert$
		PAL	$f_1(r;\theta_1)$, $f_2(r, t;\theta_2)$	$\lvert \mathbf{z}(t=0) + \int_0^1 dt(\mathbf{F}_1(\mathbf{z}) + \mathbf{F}_2(\mathbf{z}, t)) - \mathbf{z}(t=1)\rvert$, $\lvert f_2(\mathbf{z}, t;\theta_2)\rvert$

B Symbolic Regression Details

Neural network architecture: In the symbolic regression experiment, all the neural networks are fully-connected networks, which have 1 or 2 input neurons (depends on the number of input variables), 2 hidden layers (width $= 256$) with LeakyReLU ($\alpha = 0.2$), and a single output neuron.

Additive separability: The function $f_0(x_1, x_2) = (x_1^2 + x_2^2) + x_1 x_2$ can be decomposed into the additively separable part $x_1^2 + x_2^2$ and a violation part $x_1 x_2$.

We employ PAL To obtain the decomposition numerically first. we train three neural networks, $f_1(x_1;\theta_1)$, $f_2(x_2;\theta_2)$, and $f_{12}(x_1, x_2;\theta_{12})$, with $\lambda = 0.2$ for 200 epochs. We use the ADAM optimizer and annealed learning rate schedule i.e., $\{10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$ each learning rate for 50 epochs. We show three neural networks succeed in learning the ground truth decomposition in Figure 3(a)-(b). After PAL training, we then apply AI Feynman to explain the outputs i.e., symbolic expression of these three neural networks: $f_1 = x_1^2 - 0.02$, $f_2 = x_2^2 - 0.01$, and $f_{12} = x_1 x_2 + 0.03$.

Phase transition A recent work [8] theoretically proves that $L_1, L_2 \sim \lambda$ has a phase transition behavior at $\lambda = 1$ [3]. $\lambda > 1$ is the undesireable phase, so in principle one can simply choose any $\lambda < 1$ to obtain the correct result. Their numerical results suggest that $\lambda \in [0.02, 0.5]$ produce very accurate and robust results. We verify these observations in the current example.

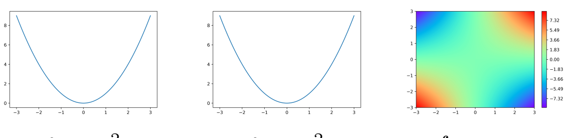
We sweep the λ region and obtained final losses (L_1 and L_2) as a function of λ in PAL and PIL, by testing $\lambda = \{0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1, 2, 5, 10, 20, 50, 100\}$. We show the results in Figure 3(c)-(d). For PAL, there is a sharp phase transition at $\lambda = 1$ and two losses are quite robust for any $\lambda < 1$. Intriguingly, the sharp phase transition for PIL has never been reported let alone studied before (to the best of knowledge), which will be interesting future directions.

Rotational invariance: We use PAL to decompose the function $f_0(x_1, x_2) = \frac{1}{2}(x_1^2 + x_2^2) + ax_1$ into the rotational-invariant part $\frac{1}{2}(x_1^2 + x_2^2)$ and violation part ax_1 ($a = 0.32$). In the experiment, we take $\lambda = 0.2$ and jointly train $f_1(R = \sqrt{x_1^2 + x_2^2}; \theta_1)$ and $f_2(x_1, x_2; \theta_2)$ for 200 epochs. We use the ADAM optimizer and annealed learning rate schedule i.e., $\{10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$ each learning rate for 50 epochs.

We apply AI Feynman to explain f_1 and f_2 parameterized by neural networks: AI Feynman discovers that $f_1 = 0.5(x_1^2 + x_2^2)$ and $f_2 = 0.31998x_1$.

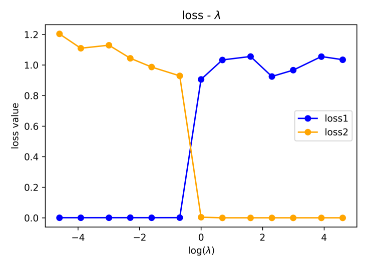
Positive Definiteness: Given a function $f_0(x) = \sin(\sin(x)) + 0$, we train the the first neural network in a nested form $f_1(f_1(x; \theta_1); \theta_1)$ for the positivity part and the second neural network for violation part. We jointly train two networks for 2000 epochs with the ADAM optimizer. We employ annealed learning rate schedule i.e., $\{10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$ each learning rate for 500 epochs. AI Feynman discovers the symbolic expression for two networks: $f_1 = -\sin(x) + 0.004$ and $f_2 = 0$.

PIL for additive separability and rotational invariance: To compare PAL and PIL, we also use PIL to learn the additive separability and rotational invariance under the same setting. The results of symbolic regression are $x_1^2 + x_2^2 + x_1x_2$ (additive separability) and $0.5(x_1^2 + x_2^2) + 0.32x_1$ (rotational invariance). Although PIL can obtain correctly the whole symbolic expression, it does not support decompositions into the property part and the violation part. For the positivity example, PIL is even not applicable since positivity is non-discriminative.

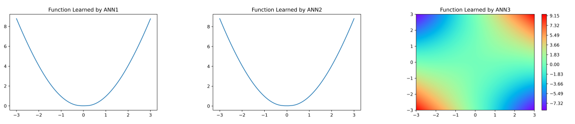


$f_1 = x_1^2$ $\quad$ $f_2 = x_2^2$ $\quad$ $f_3 = x_1x_2$

(a) Ground Truth

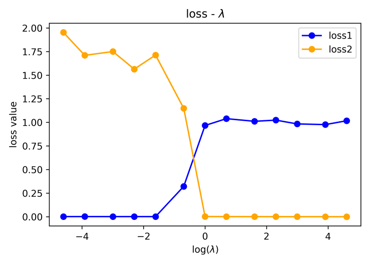


(c) PAL



$f_1(x_1; \theta_1)$ $\quad$ $f_2(x_2; \theta_2)$ $\quad$ $f_3(x_1, x_2; \theta_3)$

(b) Neural Networks Learned



(d) PIL

Figure 3: (a)(b) Additive separability learned by PAL; (c)(d) λ-loss relations in PAL and PIL.

[3]Loss functions L_1, L_2 should be defined as norms to produce the sharp phase transition behavior, e.g., mean-absolute error (MAE) or square root of mean-squared error (MSE). By contrast, MSE does not produce any sharp phase transition

C N-body Dynamics Details

Physical Model

We consider a 2D N-particle system described by their positions $\mathbf{x}_i$ and velocities $\mathbf{v}_i$ $(i = 1, \cdots, N)$. They all have unit mass $m = 1$ and have pairwise forces i.e., particle i exerts to particle j a force $\mathbf{f}_{ij} = f(r_{ij})\frac{\mathbf{r}_{ij}}{r_{ij}}$ where $\mathbf{r}_{ij} \equiv \mathbf{x}_i - \mathbf{x}_j$, $r_{ij} \equiv |\mathbf{r}_{ij}|_2$ and $f(\cdot)$ is the same for all pairs.

The dynamical equations for the j-th particle is:

$$\frac{d}{dt}\begin{pmatrix}\mathbf{x}_j\\ \mathbf{v}_j\end{pmatrix} = \begin{pmatrix}\mathbf{v}_j\\ \sum_{i\neq j}\mathbf{f}_{ij}\end{pmatrix} \tag{1}$$

Concatenating all the particle states together defines $\mathbf{z} = [\mathbf{x}_1, \cdots, \mathbf{x}_N, \mathbf{v}_1, \cdots, \mathbf{v}_N]$

$$\frac{d}{dt}\mathbf{z} \equiv \frac{d}{dt}\begin{pmatrix}\mathbf{x}_1\\ \cdots\\ \mathbf{x}_N\\ \mathbf{v}_1\\ \cdots\\ \mathbf{v}_N\end{pmatrix} = \begin{pmatrix}\mathbf{v}_1\\ \cdots\\ \mathbf{v}_N\\ \sum_{i\neq 1}\mathbf{f}_{i1}\\ \cdots\\ \sum_{i\neq N}\mathbf{f}_{iN}\end{pmatrix} \equiv \mathbf{F}(\mathbf{z}; f) \tag{2}$$

Discretizing Eq. (2) in time gives

$$\mathbf{z}^{l+1} = \mathbf{z}^l + \mathbf{F}(\mathbf{z}; f)\Delta t \quad (\mathbf{z}^l \equiv \mathbf{z}(t = l\Delta t)) \tag{3}$$

In our simulation, the initial position and initial velocity of each particle are randomly sampled from a standard Gaussian distribution $\mathbf{x}_i \sim \mathcal{N}([0,0]^T, \sigma^2\mathbf{I}_{2\times 2})$ where $\sigma = 1$. The force between each pair of two bodies is set as $f(r) = r^2$. We use Newton forward scheme as in Eq. (3) to simulate the 5-body system for 50 steps with step size $\Delta t = 0.02$. The initial positions for the 5 bodies are $(1.62, -0.61)$, $(-0.53, -1.07)$, $(0.87, -2.30)$, $(1.74, -0.76)$, $(0.32, -0.25)$. The initial velocities for the 5 bodies are $(2.92, -4.12)$, $(-0.64, -0.77)$, $(2.27, -2.20)$, $(-0.34, -1.76)$, $(0.08, 1.17)$.

Neural network architecture

Note that our goal is to infer the interaction $f(\cdot)$ based on solely the initial $\mathbf{z}(t = 0)$ and final $\mathbf{z}(t = 1)$. Our network is effectively a residual network with 50 blocks – it simply implements the computations as in Eq. (3), with only f parameterized by neural networks. To impose f to be time-independent, PAL and PIL employ different strategies.

PAL: explicit decomposition $f(r,t) = f_1(r;\theta_1) + f_2(r,t;\theta_2)$ and penalize $|f_2(r,t;\theta_2)|$. Both f_1 and f_2 are 2-hidden-layer fully-connected networks (tanh activation) with each layer containing $(1/2, 200, 200, 1)$ neurons. we take $\lambda_1 = 1$, $\lambda_2 = 0.25$, and $\lambda_3 = 1$, where λ_1 is the weight for the final position prediction loss, λ_2 is the weight for the final velocity prediction loss, and λ_3 penalty coefficient for $f_2(r,t;\theta_2)$. We train two neural networks $f_1(r;\theta_1)$ and $f_2(r,t;\theta_2)$, each repeated 100 times and stacked vertically to obtain a 100-layer Resnet. Both f_1 and f_2 are two-hidden-layer fully-connected networks (Activation function: tanh) with each layer containing $(1/2, 200, 200, 1)$ neurons. We train two networks jointly for 2000 epochs with the Adam optimizer with the learning rate 0.001.

PIL: no explicit decomposition but penalize $|f(r,t_1) - f(r,t_2)|$ for $t_1 \neq t_2$. Each block is a 2-hidden-layer fully-connected network (tanh activation) with each layer containing $(1/2, 200, 200, 1)$ neurons. For the loss function, we take $\lambda_1 = 1$, $\lambda_2 = 0.25$, and $\lambda_3 = 0.1$, where λ_1 is the weight for the final position prediction loss, λ_2 is the final velocity prediction loss, and λ_3 is the time-independence penalty loss. The time-independence loss is simply implemented as the MSE between the outputs of two neighboring blocks of the Resnet for the same input. We train the Resnet for 2000 epochs with an Adam optimizer with the learning rate 0.0001.