MIT Open Access Articles

Toward an artificial intelligence physicist for unsupervised learning

Massachusetts Institute of Technology

Toward an artificial intelligence physicist for unsupervised learning

Tailin Wu[*] and Max Tegmark

Department of Physics and Center for Brains, Minds, and Machines, Massachusetts Institute of Technology,
Cambridge, Massachusetts 02139, USA
and Theiss Research, La Jolla, California 92037, USA

We investigate opportunities and challenges for improving unsupervised machine learning using four common strategies with a long history in physics: divide and conquer, Occam's razor, unification, and lifelong learning. Instead of using one model to learn everything, we propose a paradigm centered around the learning and manipulation of *theories*, which parsimoniously predict both aspects of the future (from past observations) and the domain in which these predictions are accurate. Specifically, we propose a generalized mean loss to encourage each theory to specialize in its comparatively advantageous domain, and a differentiable description length objective to downweight bad data and "snap" learned theories into simple symbolic formulas. Theories are stored in a "theory hub," which continuously unifies learned theories and can propose theories when encountering new environments. We test our implementation, the toy "artificial intelligence physicist" learning agent, on a suite of increasingly complex physics environments. From unsupervised observation of trajectories through worlds involving random combinations of gravity, electromagnetism, harmonic motion, and elastic bounces, our agent typically learns faster and produces mean-squared prediction errors about a billion times smaller than a standard feedforward neural net of comparable complexity, typically recovering integer and rational theory parameters exactly. Our agent successfully identifies domains with different laws of motion also for a nonlinear chaotic double pendulum in a piecewise constant force field.

I. INTRODUCTION

A. Motivation

The ability to predict, analyze, and parsimoniously model observations is not only central to physics, but also a goal of unsupervised machine learning, which is a key frontier in artificial intelligence (AI) research [1]. Despite impressive recent progress with artificial neural nets, they still get frequently outmatched by human researchers at such modeling, suffering from two drawbacks:

(1) Different parts of the data are often generated by different mechanisms in different contexts. A big model that tries to fit all the data in one environment may therefore underperform in a new environment where some mechanisms are replaced by new ones, being inflexible and inefficient at combinatorial generalization [2].

(2) Big models are generally hard to interpret, and may not reveal succinct and universal knowledge such as Newton's law of gravitation that explains only some aspects of the data. The pursuit of "intelligible intelligence" in place of inscrutable black-box neural nets is important and timely, given the growing interest in AI interpretability from AI users and policymakers, especially for AI components involved in decisions and infrastructure where trust is important [3–6].

To address these challenges, we will borrow from physics the core idea of a *theory*, which parsimoniously predicts both aspects of the future (from past observations) and also the domain in which these predictions are accurate. This suggests

an alternative to the standard machine-learning paradigm of fitting a single big model to all the data: instead, learning small theories one by one, and gradually accumulating and organizing them. This paradigm suggests the four specific approaches summarized in Table I, which we combine into a toy "AI physicist" learning agent: To find individual theories from complex observations, we use the divide-and-conquer strategy with multiple theories and a generalized mean loss that encourages each theory to specialize in its own domain by giving larger gradients for better-performing theories. To find simple theories that avoid overfitting and generalize well, we use the strategy known as Occam's razor, favoring simple theories that explain a lot, using a computationally efficient approximation of the minimum-description-length (MDL) formalism. To unify similar theories found in different environments, we use the description length for clustering and then learn a "master theory" for each class of theories. To accelerate future learning, we use a lifelong learning strategy where learned theories are stored in a theory hub for future use.

B. Goals and relation to prior work

The goal of the AI physicist learning agent presented in this paper is quite limited, and does not even remotely approach the ambition level of problem solving by human physicists. The latter is likely to be almost as challenging as artificial general intelligence, which most AI researchers guess remains decades away [7,8]. Rather, the goal of this paper is to take a very modest but useful step in that direction, combining the four physics-inspired strategies above.

---
[*]tailin@mit.edu

TABLE I. AI physicist strategies tested.

Strategy	Definition
Divide and conquer	Learn multiple theories each of which specializes to fit *part* of the data very well
Occam's razor	Avoid overfitting by minimizing description length, which can include replacing fitted constants by simple integers or fractions
Unification	Try unifying learned theories by introducing parameters
Lifelong learning	Remember learned solutions and try them on future problems

Our approach complements other work on automatic program learning, such as neural program synthesis or induction [9–14] and symbolic program induction [15–19], and builds on prior machine-learning work on divide and conquer [20–22], network simplification [23–28], and continuous learning [29–32]. It is often said that babies are born scientists, and there is arguably evidence for use of all of these four strategies during childhood development as well [14].

There has been significant recent progress on AI approaches specifically linked to physics, including physical scene understanding [33], latent physical properties [34–36], learning physics simulators [37], physical concept discovery [38], an intuitive physics engine [39], and the "Sir Isaac" automated adaptive inference agent [40]. Our AI physicist is different and complementary in two fundamental ways that loosely correspond to the two motivations on the first page:

(1) All of these papers learn one big model to fit all the data. In contrast, the AI physicist learns many small models applicable in different domains, using the divide-and-conquer strategy.

(2) Our primary focus is not on making approximate predictions or discovering latent variables, but on near-exact predictions and complete intelligibility. From the former perspective, it is typically irrelevant if a model parameter changes by a tiny amount, but from a physics perspective, one is quite interested to learn that gravity weakens like distance to the power 2 rather than 1.99999314.

We share this focus on intelligibility with a long tradition of research on computational scientific discovery [41], including the Bacon system [42] and its successors [43], which induced physical laws from observations and which also used a divide-and-conquer strategy. Other work has extended this paradigm to support discovery of differential equation models from multivariate time series [44–47].

The rest of this paper is organized as follows. In Sec. II, we introduce the architecture of our AI physicist learning agent, and the algorithms implementing the four strategies. We present the results of our numerical experiments using a suite of physics environment benchmarks in Sec. III, and discuss our conclusions in Sec. IV, delegating supplementary technical details to a series of Appendices.

II. METHODS

Unsupervised learning of regularities in time series can be viewed as a supervised learning problem of predicting the future from the past. This paper focuses on the task of
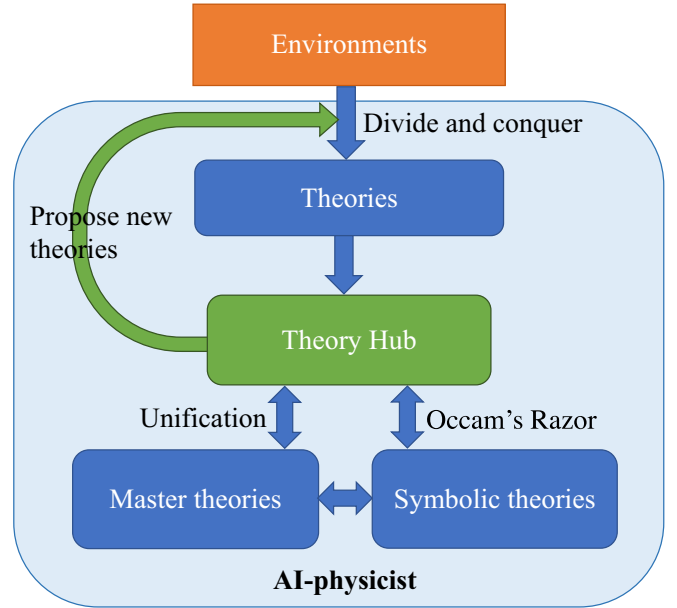

FIG. 1. AI physicist architecture.

predicting the next state vector $\mathbf{y}_t \in \mathbb{R}^d$ in a sequence from the concatenation $\mathbf{x}_t = (\mathbf{y}_{t-T}, \ldots, \mathbf{y}_{t-1})$ of the last T vectors. However, our AI physicist formalism applies more generally to learning any function $\mathbb{R}^M \mapsto \mathbb{R}^N$ from examples. In the following we first define *theory*, then introduce a unified AI physicist architecture implementing the four aforementioned strategies.

A. Definition of theory

A theory $\mathcal{T}$ is a 2-tuple $(\mathbf{f}, c)$, where $\mathbf{f}$ is a prediction function that predicts $\mathbf{y}_t$ when $\mathbf{x}_t$ is within the theory's domain, and c is a domain subclassifier which takes $\mathbf{x}_t$ as input and outputs a logit of whether $\mathbf{x}_t$ is inside this domain. When multiple theories are present, the subclassifier c's outputs are concatenated and fed into a softmax function, producing probabilities for which theory is applicable. Both $\mathbf{f}$ and c can be implemented by a neural net or symbolic formula, and can be set to learnable during training and fixed during prediction or validation.

This definition draws inspirations from physics theories (conditional statements), such as "a ball not touching anything (*condition*) with vertical velocity and height (v_0, h_0) will a time t later have $\mathbf{y} \equiv (v, h) = (v_0 - gt, h_0 + v_0 t - gt^2/2)$ (*prediction function*)." For our AI physicist, theories constitute its "atoms" of learning, as well as the building blocks for higher-level manipulations.

B. AI physicist architecture overview

Figure 1 illustrates the architecture of the AI physicist learning agent. At the center is a theory hub which stores the learned and organized theories. When encountering a new environment, the agent first inspects the hub and proposes old theories that help account for parts of the data as well as randomly initialized new theories for the rest of the data. All these theories are trained via our divide-and-conquer strategy, first jointly with our generalized mean loss then separately to fine-tune each theory in its domain (Sec. II C). Successful

theories along with the corresponding data are added to the theory hub.

The theory hub has two organizing strategies: (1) Applying Occam's razor, it snaps the learned theories, in the form of neural nets, into simpler symbolic formulas (Sec. II D). (2) Applying unification, it clusters and unifies the symbolic theories into master theories (Sec. II E). The symbolic and master theories can be added back into the theory hub, improving theory proposals for new environments. The detailed AI physicist algorithm is presented in a series of appendices. It has polynomial time complexity, as detailed in Appendix F.

C. Divide and conquer

Conventionally, a function $\mathbf{f}$ mapping $\mathbf{x}_t \mapsto \mathbf{y}_t$ is learned by parametrizing $\mathbf{f}$ by some parameter vector $\boldsymbol{\theta}$ that is adjusted to minimize a loss (empirical risk)

$$\mathcal{L} \equiv \sum_t \ell[\mathbf{f}(\mathbf{x}_t), \mathbf{y}_t], \tag{1}$$

where ℓ is some non-negative distance function quantifying how far each prediction is from the target, typically satisfying $\ell(\mathbf{y}, \mathbf{y}) = 0$. In contrast, a physicist observing an unfamiliar environment does typically *not* try to predict everything with one model, instead starting with an easier question: is there any part or aspect of the world that can be described? For example, when Galileo famously tried to model the motion of swinging lamps in the Pisa cathedral, he completely ignored everything else, and made no attempts to simultaneously predict the behavior of sound waves, light rays, weather, or subatomic particles. In this spirit, we allow multiple competing theories $\mathcal{T} = \{\mathcal{T}_i\} = \{(\mathbf{f}_i, c_i)\}$, $i = 1, 2, \ldots, M$, to specialize in different domains, with a generalized mean loss

$$\mathcal{L}_\gamma \equiv \sum_t \left\{ \frac{1}{M} \sum_{i=1}^M \ell[\mathbf{f}_i(\mathbf{x}_t), \mathbf{y}_t]^\gamma \right\}^{1/\gamma}. \tag{2}$$

When $\gamma < 0$, the loss $\mathcal{L}_\gamma$ will be dominated by whichever prediction function $\mathbf{f}_i$ fits each data point best. This dominance is controlled by γ, with $\mathcal{L}_\gamma \to \min_i \ell[\mathbf{f}_i(\mathbf{x}_t), \mathbf{y}_t]$ in the limit where $\gamma \to -\infty$. This means that the best way to minimize $\mathcal{L}_\gamma$ is for each $\mathbf{f}_i$ to specialize by further improving its accuracy for the data points where it already outperforms the other theories. The following Theorem 1 formalizes the above intuition, stating that under mild conditions for the loss function $\ell(\cdot, \cdot)$, the generalized mean loss gives larger gradient with respect to the error $|\hat{\mathbf{y}}_t - \mathbf{y}_t|$ for theories that perform better, so that a gradient-descent loss minimization encourages specialization.

Theorem 1. Let $\hat{\mathbf{y}}_t^{(i)} \equiv \mathbf{f}_i(\mathbf{x}_t)$ denote the prediction of the target $\mathbf{y}_t$ by the function $\mathbf{f}_i$, $i = 1, 2, \ldots, M$. Suppose that $\gamma < 0$ and $\ell(\hat{\mathbf{y}}_t, \mathbf{y}_t) = \ell(|\hat{\mathbf{y}}_t - \mathbf{y}_t|)$ for a monotonically increasing function $\ell(u)$ that vanishes on $[0, u_0]$ for some $u_0 \geqslant 0$, with $\ell(u)^\gamma$ differentiable and strictly convex for $u > u_0$. Then if $0 < \ell(\hat{\mathbf{y}}_t^{(i)}, \mathbf{y}_t) < \ell(\hat{\mathbf{y}}_t^{(j)}, \mathbf{y}_t)$, we have

$$\left| \frac{\partial \mathcal{L}_\gamma}{\partial u_t^{(i)}} \right| > \left| \frac{\partial \mathcal{L}_\gamma}{\partial u_t^{(j)}} \right|, \tag{3}$$

where $u_t^{(i)} \equiv |\hat{\mathbf{y}}_t^{(i)} - \mathbf{y}_t|$.

Appendix G gives the proof, and also shows that this theorem applies to mean-squared-error (MSE) loss $\ell(u) = u^2$, mean-absolute-error loss $\ell(u) = |u|$, Huber loss, and our description-length loss from the next section.

We find empirically that the simple choice $\gamma = -1$ works quite well, striking a good balance between encouraging specialization for the best theory and also giving some gradient for theories that currently perform slightly worse. We term this choice $\mathcal{L}_{-1}$ the "harmonic loss," because it corresponds to the harmonic mean of the losses for the different theories. Based on the harmonic loss, we propose an unsupervised differentiable divide-and-conquer (DDAC) algorithm (Algorithm 2 in Appendix B) that simultaneously learns prediction functions $\{\mathbf{f}_i\}$ and corresponding domain classifiers $\{c_i\}$ from observations.

Our DDAC method's combination of multiple prediction modules into a single prediction is reminiscent of AdaBoost [48]. While AdaBoost gradually upweights those modules that best predict *all* the data, DDAC instead identifies complementary modules that each predict some *part* of the data best, and encourages these modules to simplify and improve by specializing on these respective parts.

D. Occam's razor

The principle of Occam's razor, that simpler explanations are better, is quite popular among physicists. This preference for parsimony helped dispense with phlogiston, aether, and other superfluous concepts.

Our method therefore incorporates the minimum-description-length (MDL) formalism [23,26], which provides an elegant mathematical implementation of Occam's razor. It is rooted in Solomonoff's theory of inference [49] and is linked to Hutter's AIXI approach to artificial general intelligence [50]. The description length (L_d) of a data set D is defined as the number of bits required to describe it. For example, if regularities are discovered that enable data compression, then the corresponding description length is defined as the number of bits of the program that produces D as its output (including both the code bits and the compressed data bits). In our context of predicting a time series, this means that the description length is the number of bits required to describe the theories used plus the number of bits required to store all prediction errors. Finding the optimal data compression and hence computing the MDL is a famous hard problem that involves searching an exponentially large space, but any discovery reducing the description length is a step in the right direction, and provably avoids the overfitting problem that plagues many alternative machine-learning strategies [23,26].

The end goal of the AI physicist is to discover theories $\mathcal{T}$ minimizing the total description length, given by

$$L_d(\mathcal{T}, D) = L_d(\mathcal{T}) + \sum_t L_d(\mathbf{u}_t), \tag{4}$$

where $\mathbf{u}_t = \hat{\mathbf{y}}_t - \mathbf{y}_t$ is the prediction error at time step t. By discovering simple theories that can each account for parts of the data very well, the AI physicist strives to make both $L_d(\mathcal{T})$ and $\sum_t L_d(\mathbf{u}_t)$ small.
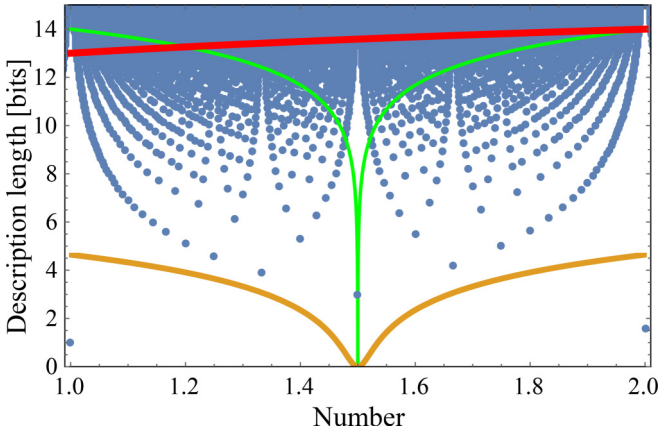
FIG. 2. The description length L_d is shown for real numbers p with $\epsilon = 2^{-14}$ (rising curve) and for rational numbers (dots). Occam's razor favors lower L_d, and our MDL rational approximation of a real parameter p is the lowest point after taking these "model bits" specifying the approximate parameter and adding the "data bits" $\mathcal{L}$ required to specify the prediction error made. The two symmetric curves illustrate the simple example where $\mathcal{L} = \log_+(\frac{x - x_0}{\epsilon})$ for $x_0 = 1.4995$, $\epsilon = 2^{-14}$ and 0.02, respectively.

Physics has enjoyed great success in its pursuit of simpler theories using rather vague definitions of simplicity. In the this spirit, we choose to compute the description length L_d not exactly, but using an approximate heuristic that is numerically efficient, and significantly simpler than more precise versions such as [51], paying special attention to rational numbers since they appear in many physics theories. We compute L_d for both theories $\mathcal{T}$ and prediction errors $\mathbf{u}_t$ as the sum of the description length of all numbers that specify them, using the following conventions for L_d for integers, rational numbers, and real numbers. Our MDL implementation differs from popular machine-learning approaches whose goal is efficiency and generalizability [27,52,53] rather than intelligibility.

The number of binary digits required to specify a natural number $n = 1, 2, 3, \dots$ is approximately $\log_2 n$, so we define $L_d(n) \equiv \log_2 n$ for natural numbers. For an integer m, we define

$$L_d(m) \equiv \log_2(1 + |m|). \tag{5}$$

For a rational number $q = m/n$, the description length is the sum of that for its integer numerator and (natural number) denominator, as illustrated in Fig. 2:

$$L_d\left(\frac{m}{n}\right) = \log_2[(1 + |m|)n]. \tag{6}$$

For a real number r and a numerical precision floor ϵ, we define

$$L_d(r) = \log_+\left(\frac{r}{\epsilon}\right), \tag{7}$$

where the function

$$\log_+(x) \equiv \tfrac{1}{2}\log_2(1 + x^2) \tag{8}$$

is plotted in Fig. 2. Since $\log_+(x) \approx \log_2 x$ for $x \gg 1$, $L_d(r)$ is approximately the description length of the integer closest to r/ϵ. Since $\log_+(x) \propto x^2$ for $x \ll 1$, $L_d(r)$ simplifies to a

quadratic (mean-squared-error) loss function below the numerical precision, which will prove useful below.[1]

Note that as long as all prediction absolute errors $|u_i| \gg \epsilon$ for some data set, minimizing the total description length $\sum_i L_d(u_i)$ instead of the MSE $\sum_i u_i^2$ corresponds to minimizing the geometric mean instead of the arithmetic mean of the squared errors, which encourages focusing more on improving already well-fitted points. $\sum_i L_d(u_i)$ drops by 1 bit whenever one prediction error is halved, which is can typically be achieved by fine-tuning the fit for many valid data points that are already well predicted while increasing L_d for bad or extraneous points at most marginally.

For numerical efficiency, our AI physicist minimizes the description length of Eq. (4) in two steps: (1) All model parameters are set to trainable real numbers, and the DDAC algorithm is applied to minimize the harmonic loss $\mathcal{L}_{-1}$ with $\ell(\mathbf{u}) \equiv \sum_i L_d(u_i)$ using Eq. (7) and the annealing procedure for the precision floor described in Appendix B. (2) Some model parameters are replaced by rational numbers as described below, followed by reoptimization of the other parameters. The idea behind the second step is that if a physics experiment or neural net training produces a parameter $p = 1.4999917$, it would be natural to interpret this as a hint, and to check if $p = 3/2$ gives an equally acceptable fit to the data, reducing total L_d. We implement step 2 using continued fraction expansion as described in Appendix C and illustrated in Fig. 3.

E. Unification

Physicists aspire not only to find simple theories that explain aspects of the world accurately, but also to discover underlying similarities between theories and *unify* them. For example, when James Clerk Maxwell corrected and unified four key formulas describing electricity and magnetism into his eponymous equations ($d\mathbf{F} = 0$, $d \star \mathbf{F} = \mathbf{J}$ in differential form notation), he revealed the nature of light and enabled the era of wireless communication.

Here we make a humble attempt to automate part of this process. The goal of the unification is to output a master theory $\mathscr{T} = \{(\mathbf{f_p}, \cdot)\}$, such that varying the parameter vector $\mathbf{p} \in \mathbb{R}^n$ can generate a continuum of theories $(\mathbf{f_p}, \cdot)$ including previously discovered ones. For example, Newton's law of gravitation can be viewed as a master theory unifying the gravitational force formulas around different planets by introducing a parameter p corresponding to planet mass. Einstein's special relativity can be viewed as a master theory unifying the approximate formulas for $v \ll c$ and $v \approx c$ motion.

We perform unification by first computing the description length $\mathrm{dl}^{(i)}$ of the prediction function $\mathbf{f}_i$ (in symbolic form) for each theory i and performing clustering on $\{\mathrm{dl}^{(i)}\}$. Unification is then achieved by discovering similarities and variations between the symbolic formulas in each cluster, retaining the similar patterns, and introducing parameters in place of the parameters that vary as detailed in Appendix D.

---

[1]Natural alternative definitions of $\log_+(x)$ include $\log_2(1 + |x|)$, $\log_2 \max(1, |x|)$, $(\ln 2)^{-1}\sinh^{-1}|x|$, and $(2\ln 2)^{-1}\sinh^{-1}(x^2)$. Unless otherwise specified, we choose $\epsilon = 2^{-32}$ in our experiments.
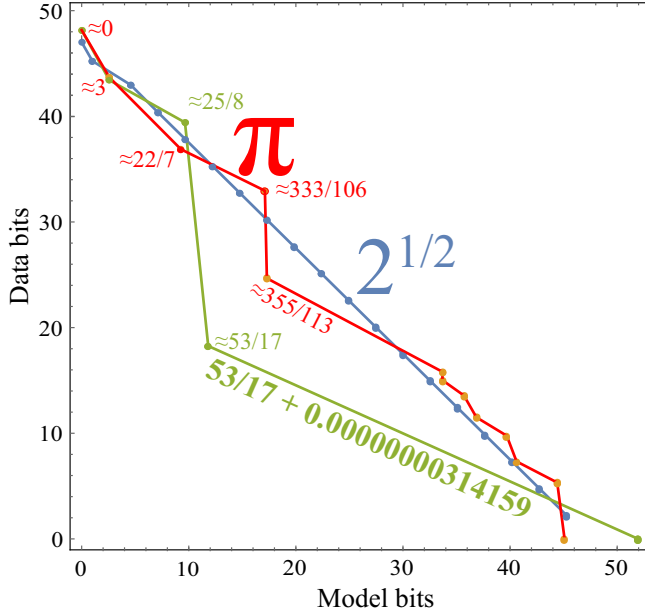
FIG. 3. Illustration of our minimum-description-length (MDL) analysis of the parameter vector $\mathbf{p} = \{\pi, \sqrt{2}, 3.43180632382353\}$. We approximate each real number r as a fraction a_k/b_k using the first k terms of its continued fraction expansion, and for each integer $k = 1, \ldots$, we plot the number of "data bits" required to encode the prediction error $r - a_k/b_k$ to 14 decimal places versus the number of "model bits" required to encode the rational approximation a_k/b_k, as described in the text. We then select the point with smallest bit sum (farthest down and left from the diagonal) as our first approximation candidate to test. Generic irrational numbers are incompressible; the total description length (model bits + data bits) is roughly independent of k as is seen for π and $\sqrt{2}$, corresponding to a line of slope -1 around which there are small random fluctuations. In contrast, the green (light gray) curve (bottom) is for a parameter that is anomalously close to a rational number, and the curve reveals this by the approximation 53/17 reducing the total description length (model+data bits) by about 16 bits.

F. Lifelong learning

Isaac Newton once said, "If I have seen further it is by standing on the shoulders of giants," emphasizing the utility of building on past discoveries. At a more basic level, our past experiences enable us humans to model new environments much faster than if we had to reacquire all our knowledge from scratch. We therefore embed a lifelong learning strategy into the architecture of the AI physicist. As shown in Fig. 1 and Algorithm 1, the theory hub stores successfully learned theories, organizes them with our Occam's razor and unification algorithms (reminiscent of what humans do while dreaming and reflecting), and when encountering new environments, uses its accumulated knowledge to propose new theories that can explain parts of the data. This both ensures that past experiences are not forgotten and enables faster learning in novel environments. The detailed algorithms for proposing and adding theories are in Appendix E.
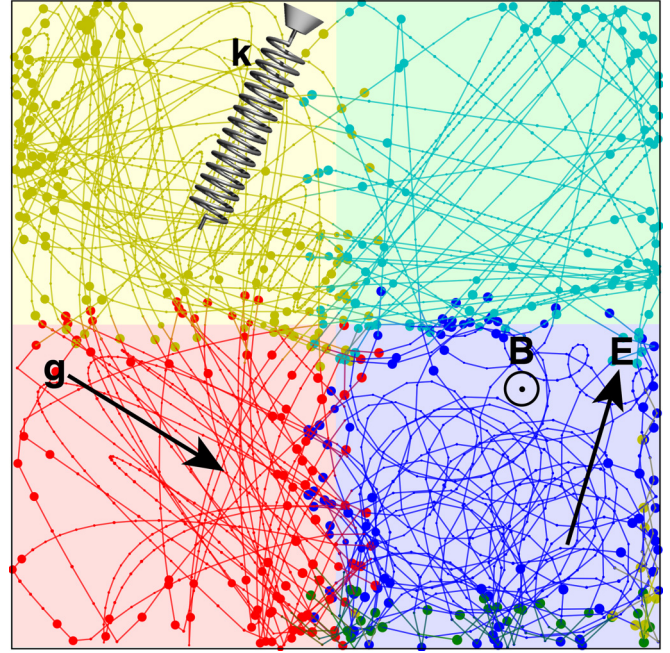

FIG. 4. In this sample mystery world, a ball moves through a harmonic potential (upper left quadrant), a gravitational field (lower left), and an electromagnetic field (lower right quadrant) and bounces elastically from four walls. The only input to the AI physicist is the sequence of dots (ball positions); the challenge is to learn all boundaries and laws of motion (predicting each position from the previous two). The color of each dot represents the domain into which it is classified by $\mathbf{c}$, and its area represents the description length of the error with which its position is predicted ($\epsilon = 10^{-6}$) after the DDAC (differentiable divide-and-conquer) algorithm; the AI physicist tries to minimize the total area of all dots.

III. RESULTS OF NUMERICAL EXPERIMENTS

A. Physics environments

We test our algorithms on two suites of benchmarks, each with increasing complexity. In all cases, the goal is to predict the two-dimensional motion as accurately as possible. One suite involves chaotic and highly nonlinear motion of a charged double pendulum in two adjacent electric fields. The other suite involves balls affected by gravity, electromagnetic fields, springs, and bounce boundaries, as exemplified in Fig. 4. Within each spatial region, the force corresponds to a potential energy function $V \propto (ax + by + c)^n$ for some constants a, b, c, where $n = 0$ (no force), $n = 1$ (uniform electric or gravitational field), $n = 2$ (spring obeying Hooke's law) or $n = \infty$ (ideal elastic bounce), and optionally involves also a uniform magnetic field. The environments are summarized in Table IV.

B. Numerical results

In the mystery world example of Fig. 4, after the DDAC algorithm 2 taking the sequence of coordinates as the only input, we see that the AI physicist has learned to simultaneously predict the future position of the ball from the previous

two, and classify without external supervision the observed inputs into four big physics domains. The predictions are seen to be more accurate deep inside the domains (tiny dots) than near boundaries (larger dots) where transitions and bounces create small domains with laws of motion that are harder to infer because of complexity and limited data. Because these small domains can be automatically inferred and eliminated once the large ones are known as described in Appendix H,

$$\hat{\mathbf{y}}_t = \begin{pmatrix} -0.99999994 & 0.00000006 & 1.99999990 & -0.00000012 \\ -0.00000004 & -1.0000000 & 0.00000004 & 2.00000000 \end{pmatrix} \mathbf{x}_t + \begin{pmatrix} 0.01088213 \\ -0.00776199 \end{pmatrix}, \tag{9}$$

with $L_d(\mathbf{f}) = 212.7$ and $\sum_t L_d(\mathbf{u}_t) = 2524.1$. The snapping stage thereafter simplifies this to

$$\hat{\mathbf{y}}_t = \begin{pmatrix} -1 & 0 & 2 & 0 \\ 0 & -1 & 0 & 2 \end{pmatrix} \mathbf{x}_t + \begin{pmatrix} 0.010882 \\ -0.007762 \end{pmatrix}, \tag{10}$$

which has lower description length in both model bits [$L_d(\mathbf{f}) = 55.6$] and data bits [$\sum_t L_d(\mathbf{u}_t) = 2519.6$] and gets transformed to the symbolic expressions

$$\hat{x}_{t+2} = 2x_{t+1} - x_t + 0.010882,$$
$$\hat{y}_{t+2} = 2y_{t+1} - y_t - 0.007762, \tag{11}$$

where we have written the 2D position vector $\mathbf{y} = (x, y)$ for brevity. During unification (Appendix D), the AI physicist discovers multiple clusters of theories based on the description length of each theory, where one cluster has L_d ranging between 48.86 and 55.63, which it unifies into a master theory $\mathbf{f_p}$ with

$$\hat{x}_{t+2} = 2x_{t+1} - x_t + p_1,$$
$$\hat{y}_{t+2} = 2y_{t+1} - y_t + p_2, \tag{12}$$

effectively discovering a "gravity" master theory out of the different types of environments it encounters. If so desired, the difference equations (12) can be automatically generalized to the more familiar-looking differential equations

$$\ddot{x} = g_x,$$
$$\ddot{y} = g_y,$$

where $g_i \equiv p_i(\Delta t)^2$, and both the harmonic oscillator equation and Lorentz force law of electromagnetism can be analogously autoinferred from other master theories learned.

Many mystery domains in our test suite involve laws of motion whose parameters include both rational and irrational numbers. To count a domain as "solved" below, we use the very stringent requirement that any rational numbers (including integers) must be discovered *exactly*, while irrational numbers must be recovered with accuracy 10^{-4}.

We apply our AI physicist to 40 mystery worlds in sequence (Appendix I). After this training, we apply it to a suite of 40 additional worlds to test how it learns different numbers of examples. The results, which Table II summarizes using the median over worlds, are shown in Tables III and IV. For comparison, we also show results for two simpler agents with similar parameter count: a "baseline" agent consisting of a three-layer feedforward MSE-minimizing leakyReLU

all accuracy benchmarks quoted below refer to points in the large domains only.

After DDAC, the AI physicist performs Occam's razor with MDL (Algorithm 3) on the learned theories. As an example, it discovers that the motion deep inside the lower left quadrant obeys the difference equation parametrized by a learned 3-layer neural net, which after the first collapseLayer transformation simplifies to

network and a "newborn" AI physicist that has not seen any past examples and therefore cannot benefit from the lifelong learning strategy.

We see that the newborn agent outperforms baseline on all the tabulated measures, and that the AI physicist does still better. Using all data, the newborn agent and AI physicist are able to predict with mean-squared prediction error below 10^{-13}, more than nine orders of magnitude below baseline. Moreover, the newborn and AI physicist agents are able to simultaneously learn the domain classifiers with essentially perfect accuracy, without external supervision. Both agents are able to solve above 90% of all the 40 mystery worlds according to our stringent criteria.

The main advantage of the AI physicist over the newborn agent is seen to be its learning speed, attaining given accuracy levels faster, especially during the early stage of learning. Remarkably, for the subsequent 40 worlds, the AI physicist

TABLE II. Summary of numerical results, taking the median over 40 mystery environments from Table III (top part) and on 40 novel environments with varying fraction of random examples (bottom parts), where each world is run with 10 random initializations and taking the best performance. Accuracies refer to big regions only.

Benchmark	Baseline	Newborn	AI physicist
$\log_{10}$ mean-squared error	−3.89	−13.95	−13.88
Classification accuracy	67.56%	100.00%	100.00%
Fraction of worlds solved	0.00%	90.00%	92.50%
Description length for $\mathbf{f}$	11 338.7	198.9	198.9
Epochs until 10^{-2} MSE	95	83	15
Epochs until 10^{-4} MSE	6925	330	45
Epochs until 10^{-6} MSE	∞	5403	3895
Epochs until 10^{-8} MSE	∞	6590	5100
$\log_{10}$ MSE			
using 100% of data	−3.78	−13.89	−13.89
using 50% of data	−3.84	−13.76	−13.81
using 10% of data	−3.16	−7.38	−10.54
using 5% of data	−3.06	−6.06	−6.20
using 1% of data	−2.46	−3.69	−3.95
Epochs until 10^{-2} MSE			
using 100% of data	95	80	15
using 50% of data	190	152.5	30
using 10% of data	195	162.5	30
using 5% of data	205	165	30
using 1% of data	397.5	235	35

TABLE III. Results for each of our first 40 mystery world benchmarks, as described in Appendix I 1. Each number is the best out of ten trials with random initializations (using seeds 0, 30, 60, 90, 120, 150, 180, 210, 240, 270), and refers to big domains only. Based on the "unsolved domains" column, we count out of 40 worlds the percentage that baseline, newborn, and AI physicist completely solve (has unsolved domain of 0), which goes to the "fraction of worlds solved" row in Table II.

Regions	$\log_{10}$ MSE			Classification accuracy			Unsolved domains			Description length		
	Base-line	New-born	AI phys.	Base-line	New-born	AI phys.	Base-line	New-born	AI phys.	Base-line	New-born	AI phys.
Free + gravity	−4.12	−14.07	−14.08	72.88%	100.00%	100.00%	2	0	0	11310.4	59.4	73.5
Free + gravity	−4.21	−14.02	−14.04	88.59%	100.00%	100.00%	2	0	0	11271.5	60.3	60.3
Free + gravity	−3.69	−14.03	−14.03	67.65%	100.00%	100.00%	2	0	0	11364.2	60.2	41.9
Free + gravity	−4.18	−13.98	−13.98	80.98%	100.00%	100.00%	2	0	0	11341.7	60.6	57.6
Free + gravity	−4.51	−14.06	−14.07	87.66%	100.00%	100.00%	2	0	0	11289.3	5.2	59.8
Free + harmonic	−3.77	−13.99	−13.94	73.54%	100.00%	100.00%	2	0	0	11333.8	94.4	139.9
Free + harmonic	−3.60	−14.05	−13.89	66.92%	100.00%	100.00%	2	0	0	11337.4	173.0	172.8
Free + harmonic	−3.77	−14.04	−13.95	59.46%	100.00%	100.00%	2	0	0	11317.5	156.0	173.8
Free + harmonic	−5.32	−10.48	−13.14	80.29%	100.00%	100.00%	2	1	0	11219.5	91.6	90.5
Free + harmonic	−3.64	−14.00	−13.89	71.70%	100.00%	100.00%	2	0	0	11369.6	143.7	136.6
Free + EM	−3.62	−13.95	−13.96	82.77%	100.00%	100.00%	2	0	0	11397.5	142.8	284.9
Free + EM	−4.13	−13.82	−13.67	76.55%	100.00%	100.00%	2	0	0	11283.0	306.2	306.2
Free + EM	−4.03	−13.45	−13.47	74.56%	99.97%	99.97%	2	0	0	11388.1	305.9	307.9
Free + EM	−4.31	−13.77	−13.62	86.68%	99.91%	99.91%	2	0	0	11257.7	152.0	133.5
Free + EM	−4.32	−14.00	−14.05	84.55%	100.00%	100.00%	2	0	0	11258.9	303.7	303.8
Free + EM rational	−3.45	−13.96	−13.95	77.88%	99.96%	99.93%	2	0	0	11414.9	194.2	195.8
Free + EM rational	−3.90	−13.96	−13.91	71.13%	100.00%	100.00%	2	0	0	11340.0	199.0	199.0
Free + EM rational	−4.12	−13.97	−13.90	72.78%	100.00%	100.00%	2	0	0	11330.7	198.8	198.8
Free + EM rational	−4.02	−14.07	−14.00	77.92%	100.00%	100.00%	2	0	0	11323.5	197.8	197.8
Free + EM rational	−4.83	−13.87	−13.86	91.14%	100.00%	100.00%	2	0	0	11247.1	10.3	13.9
Free + gravity + harmonic	−4.08	−14.03	−13.95	60.08%	100.00%	100.00%	3	0	0	11269.0	191.8	191.9
Free + gravity + harmonic	−4.31	−14.02	−13.66	63.01%	100.00%	100.00%	3	0	0	11334.2	170.4	83.1
Free + gravity + harmonic	−4.01	−14.01	−13.99	67.48%	100.00%	100.00%	3	0	0	11351.0	168.7	198.9
Free + gravity + harmonic	−3.64	−13.97	−13.88	60.02%	99.97%	99.93%	3	0	0	11374.6	225.7	225.7
Free + gravity + harmonic	−4.11	−7.42	−7.43	51.63%	100.00%	99.97%	3	1	1	11313.7	193.5	179.2
Free + gravity + EM	−3.79	−13.93	−13.47	57.89%	100.00%	100.00%	3	0	0	11334.0	323.9	346.8
Free + gravity + EM	−4.18	−14.00	−14.00	77.16%	100.00%	100.00%	3	1	1	11301.0	277.9	96.2
Free + gravity + EM	−3.38	−13.58	−13.87	53.33%	100.00%	99.96%	3	0	0	11381.4	360.4	364.0
Free + gravity + EM	−3.46	−13.87	−13.89	49.08%	100.00%	100.00%	3	0	0	11370.1	354.0	350.4
Free + gravity + EM	−3.54	−13.69	−13.83	51.28%	100.00%	100.00%	3	0	0	11370.3	331.1	320.7
Free + harmonic + EM	−3.87	−13.82	−13.55	67.27%	100.00%	100.00%	3	0	0	11404.0	267.1	275.4
Free + harmonic + EM	−3.69	−13.87	−10.93	56.02%	99.97%	99.94%	3	0	0	11413.4	468.5	464.9
Free + harmonic + EM	−4.06	−13.39	−13.56	70.87%	100.00%	100.00%	3	0	0	11340.0	452.3	452.3
Free + harmonic + EM	−3.46	−13.94	−10.51	59.02%	99.97%	99.93%	3	0	0	11416.0	475.5	471.9
Free + harmonic + EM	−3.70	−13.75	−13.82	61.67%	100.00%	100.00%	3	0	0	11354.9	466.8	466.8
Free + gravity + harmonic + EM	−3.76	−13.82	−9.48	27.93%	100.00%	99.94%	4	0	0	11358.8	526.9	530.4
Free + gravity + harmonic + EM	−3.74	−13.00	−13.18	40.80%	100.00%	99.97%	4	1	1	11284.8	418.5	389.1
Free + gravity + harmonic + EM	−4.09	−13.97	−13.75	35.69%	100.00%	100.00%	4	0	0	11297.4	504.6	504.6
Free + gravity + harmonic + EM	−3.63	−13.80	−9.99	31.61%	100.00%	99.97%	4	0	0	11407.4	526.3	526.2
Free + gravity + harmonic + EM	−3.51	−6.37	−13.52	32.97%	100.00%	100.00%	4	0	0	11445.8	527.4	527.5
Median	−3.89	−13.95	−13.88	67.56%	100.00%	100.00%	2.5	0.00	0.00	11338.7	198.9	198.9
Mean	−3.94	−13.44	−13.29	65.51%	99.99%	99.99%	2.6	0.10	0.07	11337.9	253.7	252.9

reaches 0.01 MSE within 35 epochs using as little as 1% of the data, performing almost as well as with 50% of the data much better than the newborn agent. This illustrates that the lifelong learning strategy enables the AI physicist to learn much faster in novel environments with less data. This is much like an experienced scientist can solve new problems way faster than a beginner by building on prior knowledge about similar problems.

Our double-pendulum mysteries (Appendix I 2) are more challenging for all the agents, because the motion is more nonlinear and indeed chaotic. Although none of our double-pendulum mysteries get exactly solved according to our very stringent above-mentioned criterion, Fig. 5 illustrates that the newborn agent does a good job: it discovers the two domains and classifies points into them with an accuracy of 96.5%. Overall, the newborn agent has a median best accuracy

TABLE IV. Same as previous table, but showing number of training epochs required to reach various MSE prediction accuracies. We record the metrics every 5 epochs, so all the epochs are multiples of 5. Note that the AI physicist has superseded 10^{-2} MSE already by 0 epochs for some environments, showing that thanks to the lifelong learning strategy which proposes previously learned theories in novel environments, reasonably good predictions can sometimes be achieved even without gradient descent training.

Regions	Epochs to 10^{-2}			Epochs to 10^{-4}			Epochs to 10^{-6}			Epochs to 10^{-8}		
	Base-line	New-born	AI phys.	Base-line	New-born	AI phys.	Base-line	New-born	AI phys.	Base-line	New-born	AI phys.
Free + gravity	100	85	85	8440	120	120	∞	4175	3625	∞	6315	4890
Free + gravity	100	70	10	4680	190	35	∞	2900	4650	∞	2995	6500
Free + gravity	85	100	15	∞	135	30	∞	8205	3815	∞	9620	6455
Free + gravity	95	75	20	7495	140	25	∞	6735	1785	∞	8040	2860
Free + gravity	110	75	0	1770	295	35	∞	3740	3240	∞	7030	3460
Free + harmonic	80	75	20	∞	145	25	∞	2725	4050	∞	2830	6145
Free + harmonic	85	75	20	∞	80	25	∞	7965	1690	∞	10000	3400
Free + harmonic	95	75	30	∞	110	30	∞	1805	3895	∞	1855	3900
Free + harmonic	25	20	5	1285	460	10	∞	5390	1060	∞	7225	6385
Free + harmonic	80	95	5	∞	110	20	∞	4380	3300	∞	4800	4035
Free + EM	90	85	20	∞	1190	115	∞	6305	3380	∞	6590	3435
Free + EM	125	120	0	6240	885	70	∞	7310	1865	∞	7565	1865
Free + EM	115	115	15	15260	600	70	∞	2430	1225	∞	2845	4435
Free + EM	145	90	0	6650	140	0	∞	3000	5205	∞	4530	8735
Free + EM	80	80	10	965	200	25	∞	4635	1970	∞	4690	2870
Free + EM rational	80	75	0	∞	580	70	∞	5415	4150	∞	5445	4175
Free + EM rational	100	100	10	∞	460	45	∞	2560	965	∞	2575	5760
Free + EM rational	140	95	10	11050	455	65	∞	1960	1150	∞	6295	4005
Free + EM rational	120	100	5	13315	325	175	∞	3970	1290	∞	4335	3560
Free + EM rational	35	30	35	1155	335	35	∞	3245	2130	∞	5115	5610
Free + gravity + harmonic	150	75	25	9085	130	30	∞	3870	6145	∞	5555	6185
Free + gravity + harmonic	145	90	5	6915	140	25	∞	4525	3720	∞	10275	4430
Free + gravity + harmonic	105	100	15	6925	155	40	∞	6665	6560	∞	8915	6845
Free + gravity + harmonic	95	95	5	∞	120	30	∞	5790	10915	∞	18450	13125
Free + gravity + harmonic	135	95	15	7970	190	45	∞	13125	7045	∞	∞	∞
Free + gravity + EM	130	100	20	∞	575	40	∞	3215	5095	∞	3215	5100
Free + gravity + EM	125	110	15	5650	160	30	∞	6085	4720	∞	8025	4980
Free + gravity + EM	80	65	15	∞	630	120	∞	4100	6250	∞	4100	6570
Free + gravity + EM	80	75	5	∞	90	45	∞	5910	5815	∞	7295	6090
Free + gravity + EM	80	85	20	∞	1380	465	∞	2390	11425	∞	7450	11510
Free + harmonic + EM	85	75	25	∞	600	150	∞	3775	4525	∞	4675	5070
Free + harmonic + EM	85	90	25	∞	1245	200	∞	6225	2340	∞	6390	3180
Free + harmonic + EM	115	85	15	16600	190	35	∞	6035	1515	∞	10065	2110
Free + harmonic + EM	80	70	35	∞	720	195	∞	6990	3895	∞	6995	6115
Free + harmonic + EM	85	65	10	∞	985	165	∞	5660	1670	∞	5820	1820
Free + gravity + harmonic + EM	90	75	0	∞	540	255	∞	8320	7390	∞	9770	7590
Free + gravity + harmonic + EM	95	80	15	∞	1265	635	∞	6520	6365	∞	8475	6475
Free + gravity + harmonic + EM	130	85	10	8620	575	105	∞	6320	4035	∞	9705	7685
Free + gravity + harmonic + EM	75	80	0	∞	815	425	∞	7575	8405	∞	10440	8620
Free + gravity + harmonic + EM	80	65	20	∞	735	280	∞	6715	4555	∞	12495	8495
Median	95	83	15	∞	330	45	∞	5403	3895	∞	6590	5100
Mean	98	82	15	∞	455	109	∞	5217	4171	∞	6892	5499

of 91.0% compared with the baseline of 76.9%. The MSE prediction error is comparable to the baseline performance ($\sim 4 \times 10^{-4}$) in the median, since both architectures have similar large capacity. We analyze this challenge and opportunities for improvement below.

IV. CONCLUSIONS

We have presented a toy "AI physicist" unsupervised learning agent centered around the learning and manipulation of theories, which in polynomial time learns to parsimoniously predict both aspects of the future (from past observations) and the domain in which these predictions are accurate.

A. Key findings

Testing it on a suite of mystery worlds involving random combinations of gravity, electromagnetism, harmonic motion, and elastic bounces, we found that its divide-and-conquer and Occam's razor strategies effectively identified domains
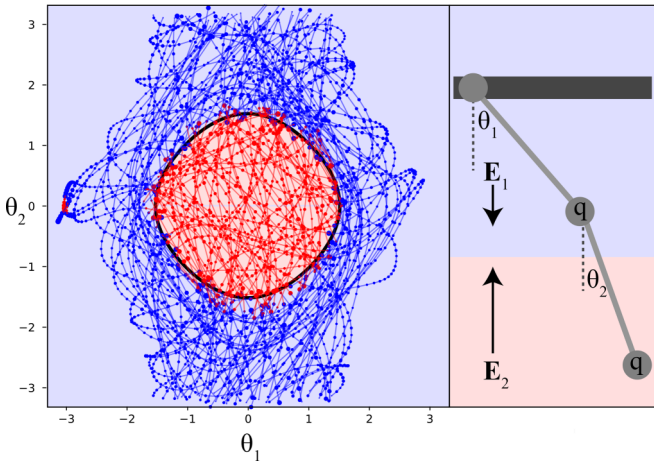
FIG. 5. In this mystery, a charged double pendulum moves through two different electric fields $\mathbf{E}_1$ and $\mathbf{E}_2$, with a domain boundary corresponding to $\cos\theta_1 + \cos\theta_2 = 1.05$ (the black curve above left, where the lower charge crosses the $\mathbf{E}$-field boundary). The color of each dot represents the domain into which it is classified by a newborn agent, and its area represents the description length of the error with which its position is predicted, for a precision floor $\epsilon \approx 0.006$. In this world, the newborn agent has a domain prediction accuracy of 96.5%.

with different laws of motion and reduced the mean-squared prediction error billionfold, typically recovering integer and rational theory parameters exactly. These two strategies both encouraged prediction functions to specialize: the former on the domains they handled best, and the latter on the data points within their domain that they handled best. Adding the lifelong learning strategy greatly accelerated learning in novel environments.

B. What has been learned?

Returning to the broader context of unsupervised learning from Sec. I raises two important questions: what is the *difficulty* of the problems that our AI physicist solved, and what is the *generality* of our paradigm?

In terms of *difficulty*, our solved physics problems are clearly on the easier part of the spectrum, so if we were to have faced the *supervised* learning problem where the different domains were pre-labeled, the domain learning would have been a straightforward classification task and the forecasting task could have been easily solved by a standard feedforward neural network. Because the real world is generally unlabeled, we instead tackled the more difficult problem where boundaries of multiple domains had to be learned concurrently with the dynamical evolution rules in a fully unsupervised fashion. The dramatic performance improvement over a traditional neural network seen in Table III reflects the power of the divide-and-conquer and Occam's razor strategies, and their robustness is indicated by the the fact that unsupervised domain discovery worked well even for the two-field nonlinear double-pendulum system whose dynamic is notoriously chaotic and whose domain boundary is the curved rhomboid $\cos\theta_1 + \cos\theta_2 = 1.05$.

In terms of *generality*, our core contribution lies in the AI physicist paradigm we propose (combining divide and con-

quer, Occam's razor, unification, and lifelong learning), not in the specific implementation details. Here we draw inspiration from the history of the Turing machine: Turing's initial implementation of a universal computer was very inefficient for all but toy problems, but his framework laid out the essential architectural components that subsequent researchers developed into today's powerful computers. What has been learned is that our AI physicist paradigm outperforms traditional deep learning on a test suite of problems even though it is a fully general paradigm that is not designed specifically for these problems. For example, it is defined to work for an arbitrary number of input spatial dimensions, spatial domains, past time steps used, boundaries of arbitrary shapes, and evolution laws of arbitrary complexity.

From the above-mentioned successes and failures of our paradigm, we have also learned about promising opportunities for improvement of the implementation which we will now discuss. First of all, the more modest success in the double-pendulum experiments illustrated the value of learned theories being *simple*: if they are highly complex, they are less likely to unify or generalize to future environments, and the correspondingly complex baseline model will have less incentive to specialize because it has enough expressive power to approximate the motion in all domains at once. It will therefore be valuable to improve techniques for simplifying complex learned neural nets. The specific implementation details for the Occam's razor toolkit would then change, but the principle and numerical objective would remain the same: reducing their total description length from Eq. (4). There are many promising opportunities for this using techniques from the Monte-Carlo-Markov-chain-based and genetic techniques [54], reinforcement learning [55,56], and symbolic regression [57,58] literature to simplify and shrink the model architecture. Also, it will be valuable and straightforward to generalize our implementation to simplify not only the prediction functions, but also the classifiers, for example to find sharp domain boundaries composed of hyperplanes or other simple surfaces.

Analogously, there are many ways in which the unification and lifelong learning toolkits can be improved while staying within our AI physicist paradigm. For example, unification can undoubtedly be improved by using more sophisticated clustering techniques for grouping the learned theories with similar ones. Lifelong learning can probably be made more efficient by using better methods for determining which previous theories to try when faced by new data, for example by training a separate neural network to perform this prediction task.

C. Outlook

In summary, these and other improvements to the algorithms that implement our AI physicist paradigm could enable future unsupervised learning agents to learn simpler and more accurate models faster from fewer examples, and also to discover accurate symbolic expressions for more complicated physical systems. More broadly, AI has recently been used with great success to tackle problems in diverse areas of physics, ranging from quantum state reconstruction [59] to phase transitions [60–62], planetary dynamics [63], and

TABLE V. Hyperparameter settings in the numerical experiments. For a fair comparison between baseline and the other agents that can have up to 4 theories, the number of neurons in each layer of baseline is larger so that the total number of parameters is roughly the same for all agents. The baseline agent in mystery worlds has leakyReLU activation to be able to account for different domains.

	Hyperparameter	Environments	Baseline	Newborn	AI physicist
γ	Generalized-mean-loss exponent	All	-1	-1	-1
$\beta_{\mathbf{f}}$	Initial learning rate for $\mathbf{f}_\theta$	All	0.005	0.005	0.005
β_c	Initial learning rate for $\mathbf{c}_\phi$	All	0.001	0.001	0.001
K	Number of gradient iterations	All	10000	10000	10000
σ_c	Hidden layer activation function in $\mathbf{c}_\phi$	All		leakyReLU	leakyReLU
N_{lay}^c	Number of layers in $\mathbf{c}_\phi$	All		3	3
C	Initial number of clusters in theory unification	All	4	4	4
ϵ_{MSE}	MSE regularization strength	All	10^{-7}	10^{-7}	10^{-7}
ϵ_{L1}	Final L_1 regularization strength	Mystery worlds	10^{-8}	10^{-8}	10^{-8}
		Double pendulum	10^{-7}	10^{-7}	10^{-7}
N_{lay}^f	Number of layers in $\mathbf{f}_\theta$	Mystery worlds	3	3	3
		Double pendulum	6	6	6
N_{neur}^f	Number of neurons in $\mathbf{f}_\theta$	Mystery worlds	16	8	8
		Double pendulum	320	160	
N_{neur}^c	Number of neurons in $\mathbf{c}_\phi$	Mystery worlds		8	8
		Double pendulum		6	
T	Maximum time horizon for input	Mystery worlds	2	2	2
		Double pendulum	1	1	1
σ_f	Hidden layer activation function in $\mathbf{f}_\theta$	Mystery worlds	leakyReLU	linear	linear
		Double pendulum	tanh	tanh	
M_0	Initial number of theories	Mystery worlds	1	2	2
		Double pendulum	1	2 and 3	
M	Maximum number of theories	Mystery worlds	1	4	4
		Double pendulum	1	2 and 3	
ϵ_{add}	MSE threshold for theory adding	Mystery worlds		2×10^{-6}	2×10^{-6}
		Double pendulum		∞	
η_{insp}	Inspection threshold for theory adding	Mystery worlds		30%	30%
		Double pendulum		∞	
η_{split}	Splitting threshold for theory adding	Mystery worlds		5%	5%
		Double pendulum		∞	
η_{del}	Fraction threshold for theory deletion	Mystery worlds		0.5%	0.5%
		Double pendulum		100%	

particle physics [64]. We hope that building on the ideas of this paper may one day enable AI to help us discover entirely novel physical theories from data.

APPENDIX A: AI PHYSICIST ALGORITHM

The detailed AI physicist algorithm is presented in Algorithm 1, with links to each of the individual subalgorithms. Like most numerical methods, the algorithm contains a number of hyperparameters that can be tuned to optimize performance; Table V lists them and their settings for our numerical experiments.

APPENDIX B: THE DIFFERENTIABLE DIVIDE-AND-CONQUER ALGORITHM

Here we elaborate on our differentiable divide-and-conquer (DDAC) algorithm with generalized mean loss $\mathcal{L}_\gamma$ [Eq. (2)]. This loss with $\gamma < 0$ works with a broad range of distance functions ℓ satisfying Theorem 1. Since the goal of our AI physicist is to minimize the overall description length L_d from Eq. (4), we choose ℓ to be the L_d loss function of

Algorithm 1. AI physicist: Overall algorithm.

Given observations $D = \{(\mathbf{x}_t, \mathbf{y}_t)\}$ from new environment:

1: $\mathcal{T}_{M_0} \leftarrow$ **Hub**.propose-theories(D, M_0) (Alg. 5)
2: $\mathcal{T} \leftarrow$ differentiable-divide-and-conquer($D, \mathcal{T}_{M_0}$)(Alg. 2)
3: **Hub**.add-theories($\mathcal{T}, D$) (Alg. 6)

Organizing theory hub:
$\mathcal{T} \leftarrow$ **Hub**.Occam's-razor-with-MDL($\mathcal{T}, D$) (Alg. 3)
$\mathcal{T} \leftarrow$ **Hub**.unify($\mathcal{T}$) (Alg. 4)

Algorithm 2. AI physicist: Differentiable divide and conquer with harmonic loss.

Require Data set $D = \{(\mathbf{x}_t, \mathbf{y}_t)\}$
Require M: number of initial total theories for training
Require $\mathcal{T}_{M_0} = \{(\mathbf{f}_i, c_i)\}, i = 1, \ldots, M_0, 0 \leqslant M_0 \leqslant M$:
theories proposed from theory hub
Require K: number of gradient iterations
Require $\beta_{\mathbf{f}}, \beta_{\mathbf{c}}$: learning rates
Require ϵ_0: initial precision floor

1: Randomly initialize $M - M_0$ theories $\mathcal{T}_i, i = M_0 + 1, \ldots, M$.
   Denote $\mathcal{T} = (\mathcal{T}_1, \ldots, \mathcal{T}_M), \mathbf{f}_{\boldsymbol{\theta}} = (\mathbf{f}_1, \ldots, \mathbf{f}_M), \mathbf{c}_{\boldsymbol{\phi}} = (c_1, \ldots, c_M)$
   with learnable parameters $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$.

// Harmonic training with L_d loss:
2: $\epsilon \leftarrow \epsilon_0$
3: **for** k **in** $\{1, 2, 3, 4, 5\}$ **do**:
4:     $\mathcal{T} \leftarrow$ **IterativeTrain**$(\mathcal{T}, D, \ell_{L_d, \epsilon}, \mathcal{L}_{-1})$, where
       $\mathcal{L}_{-1} \equiv \sum_t \left\{ \frac{1}{M} \sum_{i=1}^{M} \ell[\mathbf{f}_i(\mathbf{x}_t), \mathbf{y}_t]^{-1} \right\}^{-1}$ [Eq. (2)]
5:     $\epsilon \leftarrow$ set_epsilon$(\mathcal{T}, D)$ *// median prediction error*
6: **end for**

// Fine-tune each theory and its domain:
7: **for** k **in** $\{1, 2\}$ **do**:
8:     $\mathcal{T} \leftarrow$ **IterativeTrain**$(\mathcal{T}, D, \ell_{L_d, \epsilon}, \mathcal{L}_{\text{dom}})$, where
       $\mathcal{L}_{\text{dom}} \equiv \sum_t \ell[\mathbf{f}_{i_t}(\mathbf{x}_t), \mathbf{y}_t]$ with $i_t = \arg\max_i c_i(\mathbf{x}_t)$
9:     $\epsilon \leftarrow$ set_epsilon$(\mathcal{T}, D)$ *// median prediction error*
10: **end for**
11: **return** $\mathcal{T}$

---

subroutine IterativeTrain$(\mathcal{T}, D, \ell, \mathcal{L})$:
s1: **for** k **in** $\{1, \ldots, K\}$ **do**:
       // Gradient descent on $\mathbf{f}_{\boldsymbol{\theta}}$ with loss $\mathcal{L}$:
s2: $\mathbf{g}_{\mathbf{f}} \leftarrow \nabla_{\boldsymbol{\theta}} \mathcal{L}[\mathcal{T}, D, \ell]$
s3:   Update $\boldsymbol{\theta}$ using gradients $\mathbf{g}_{\mathbf{f}}$ (e.g., Adam [65] or SGD)

       // Gradient descent on $\mathbf{c}_{\boldsymbol{\phi}}$ with the best-performing
       *   theory index as target:*
s4: $b_t \leftarrow \arg\min_i \{\ell[\mathbf{f}_i(\mathbf{x}_t), \mathbf{y}_t]\}, \forall t$
s5: $\mathbf{g}_{\mathbf{c}} \leftarrow \nabla_{\boldsymbol{\phi}} \sum_{(\mathbf{x}_t, \cdot) \in D}$ CrossEntropy[softmax$(\mathbf{c}_{\boldsymbol{\phi}}(\mathbf{x}_t)), b_t$]
s6:   Update $\boldsymbol{\phi}$ using gradients $\mathbf{g}_{\mathbf{c}}$ (e.g., Adam [65] or SGD)
s7: **end for**
s8: $\mathcal{T} \leftarrow$ AddTheories$(\mathcal{T}, D, \ell, \mathcal{L})$ //Optional
s9: $\mathcal{T} \leftarrow$ DeleteTheories$(\mathcal{T}, D, \ell)$ //Optional
s10: **return** $\mathcal{T}$

Eq. (7) together with $\gamma = -1$ (harmonic loss), which works quite well in practice.

Algorithm 2 describes our differentiable divide-and-conquer implementation, which consists of two stages. In the first stage (steps 2–6), it applies the subroutine IterativeTrain$(\mathcal{T}, D, \ell_{L_d, \epsilon}, \mathcal{L}_{-1})$ with harmonic loss $\mathcal{L}_{-1}$ to train the theories $\mathcal{T}$ a few times with the precision floor ϵ gradually lowered according to the following annealing schedule. We set the initial precision floor ϵ to be quite large so that ℓ initially approximates an MSE loss function. After each successive iteration, we reset ϵ to the median prediction error.

The L_d loss function from Eq. (7) is theoretically desirable but tricky to train, both because it is nonconvex and because it is quite flat and uninformative far from its minimum. Our annealing schedule helps overcome both problems: initially when ϵ is large, it approximates MSE loss which is convex and guides the training to a good approximate minimum, which further training accurately pinpoints as ϵ is reduced.

The subroutine IterativeTrain forms the core of the algorithm. In the first stage (steps 2–6), it uses the harmonic mean of the L_d loss of multiple prediction functions $\mathbf{f}_{\boldsymbol{\theta}} = (\mathbf{f}_1, \ldots, \mathbf{f}_M)$ [i.e., Eq. (2) with $\gamma = -1$ and $\ell = L_d$] to simultaneously train these functions, encouraging them to each specialize in the domains where they predict best (as proven by Theorem 1), and simultaneously trains the domain classifier $\mathbf{c}_{\boldsymbol{\phi}} = (c_1, \ldots, c_M)$ using each example's best-performing prediction function as target, with categorical cross-entropy loss. After several rounds of IterativeTrain with successively lower precision floors, each prediction function typically becomes good at predicting part of the data set, and the domain classifier becomes good at predicting for each example which prediction function will predict best.

AddTheories$(\mathcal{T}, D, \ell, \mathcal{L})$ inspects each theory $\mathcal{T}_i$ describing at least a large fraction η_{insp} (we use 30%) of the examples to see if a non-negligible proportion η_{split} of examples (we use a threshold of 5%) of the examples inside its domain have MSE larger than a certain limit ϵ_{add} (we use 2×10^{-6}) and thus warrant splitting off into a separate domain.

If so, it uses those examples to initialize a new theory $\mathcal{T}_{M+1}$, and performs tentative training together with other theories using IterativeTrain without steps s8 and s9 (it is also possible to allow steps s8 and s9 in this recursive calling of IterativeTrain, which will enable a recursive adding of theories for not-well-explained data, and may enable a more powerful DDAC algorithm). If the resulting loss $\mathcal{L}$ is smaller than before adding the new theory, $\mathcal{T}_{M+1}$ is accepted and retained; otherwise it is rejected and training reverts to the checkpoint before adding the theory. DeleteTheories$(\mathcal{T}, D, \ell)$ deletes theories whose domain or best-predicted examples cover a negligible fraction of the examples (we use a delete threshold $\eta_{\text{del}} = 0.5\%$).

In the second stage (steps 7–10), the IterativeTrain is applied again, but the loss for each example $(\mathbf{x}_t, \mathbf{y}_t)$ is using only the theory that the domain classifier $\mathbf{c}_{\boldsymbol{\phi}} = (c_1, c_2, \ldots, c_M)$ predicts (having the largest logit). In this way, we iteratively fine-tune the prediction functions $\{\mathbf{f}_i\}$ with respect to each of its domain, and fine-tune the domain to the best-performing theory at each point. The reason that we assign examples to domains using our domain classifier rather than prediction accuracy is that the trained domains are likely to be simpler and more contiguous, thus generalizing better to unseen examples than, e.g., the nearest neighbor algorithm.

We now specify the default hyperparameters used for Algorithm 1 in our experiments (unless otherwise specified). We set the initial total number of theories $M = 4$, from which $M_0 = 2$ theories are proposed from the theory hub. The initial precision floor $\epsilon_0 = 10$ and the number of gradient iterations $K = 10\,000$. We use the Adam [65] optimizer with default parameters for the optimization of both the prediction function and the domain classifier. We randomly split each data set D into D_{train} and D_{test} with 4:1 ratio. The D_{test} is used only for

---

Algorithm 3. AI physicist: Occam's razor with MDL.

Require Data set $D = \{(\mathbf{x}_t, \mathbf{y}_t)\}$
Require $\mathcal{T} = \{(\mathbf{f}_i, c_i)\}, i = 1, \ldots, M$: theories trained after Alg. 2
Require ϵ: Precision floor for $\ell_{L_d, \epsilon}$
1: **for** i in $\{1, \ldots, M\}$ **do**:
2:     $D^{(i)} \leftarrow \{(\mathbf{x}_t, \mathbf{y}_t) | \arg\max_j \{c_j(\mathbf{x}_t)\} = i\}$
3:     $\mathbf{f}_i \leftarrow$ **MinimizeDL**(collapseLayers, $\mathbf{f}_i, D^{(i)}, \epsilon$)
4:     $\mathbf{f}_i \leftarrow$ **MinimizeDL**(localSnap, $\mathbf{f}_i, D^{(i)}, \epsilon$)
5:     $\mathbf{f}_i \leftarrow$ **MinimizeDL**(integerSnap, $\mathbf{f}_i, D^{(i)}, \epsilon$)
6:     $\mathbf{f}_i \leftarrow$ **MinimizeDL**(rationalSnap, $\mathbf{f}_i, D^{(i)}, \epsilon$)
7:     $\mathbf{f}_i \leftarrow$ **MinimizeDL**(toSymbolic, $\mathbf{f}_i, D^{(i)}, \epsilon$)
8: **end for**
9: **return** $\mathcal{T}$

---

subroutine MinimizeDL(transformation, $\mathbf{f}_i, D^{(i)}, \epsilon$):
s1: **while** transformation.is_applicable($\mathbf{f}_i$) **do**:
s2: $\mathrm{dl}_0 \leftarrow L_d(\mathbf{f}_i) + \sum_{(\mathbf{x}_t, \mathbf{y}_t) \in D^{(i)}} \ell_{L_d, \epsilon}[\mathbf{f}_i(\mathbf{x}_t), \mathbf{y}_t]$
s3: $f_{\text{clone}} \leftarrow \mathbf{f}_i$ // *clone* $\mathbf{f}_i$ *in case transformation fails*
s4: $\mathbf{f}_i \leftarrow$ transformation($\mathbf{f}_i$)
s5: $\mathbf{f}_i \leftarrow \text{Minimize}_{\mathbf{f}_i} \sum_{(\mathbf{x}_t, \mathbf{y}_t) \in D^{(i)}} \ell_{L_d, \epsilon}[\mathbf{f}_i(\mathbf{x}_t), \mathbf{y}_t]$
s6: $\mathrm{dl}_1 \leftarrow L_d(\mathbf{f}_i) + \sum_{(\mathbf{x}_t, \mathbf{y}_t) \in D^{(i)}} \ell_{L_d, \epsilon}[\mathbf{f}_i(\mathbf{x}_t), \mathbf{y}_t]$
s7:   **if** $\mathrm{dl}_1 > \mathrm{dl}_0$ **return** f_{clone}
s8: **end while**
s9: **return** $\mathbf{f}_i$

---

evaluation of performance. The batch size is set to min(2000, $|D_{\text{train}}|$). We set the initial learning rate $\beta_{\mathbf{f}} = 5 \times 10^{-3}$ for the prediction functions $\mathbf{f}_{\boldsymbol{\theta}}$ and $\beta_{\mathbf{c}} = 10^{-3}$ for the domain classifier $\mathbf{c}_{\boldsymbol{\phi}}$. We also use a learning rate scheduler that monitors the validation loss every 10 epochs, and divides the learning rate by 10 if the validation loss has failed to decrease after 40 monitoring points and stops training early if there is no decrease after 200 epochs—or if the entire MSE loss for all the theories in their respective domains drops below 10^{-12}.

To the main harmonic loss $\mathcal{L}_\gamma$, we add two regularization terms. One is L_1 loss whose strength increases quadratically from 0 to ϵ_{L1} during the first 5000 epochs and remains constant thereafter. The second regularization term is a very small MSE loss of strength ϵ_{MSE}, to encourage the prediction functions to remain not too far away from the target outside their domain.

APPENDIX C: OCCAM'S RAZOR WITH MDL ALGORITHM

Pushing on after the DDAC algorithm with harmonic loss that minimizes the $\sum_t L_d(\mathbf{u}_t)$ term in Eq. (4), the AI physicist then strives to minimize the $L_d(\mathcal{T})$ term, which can be decomposed as $L_d(\mathcal{T}) = L_d(\mathbf{f}_{\boldsymbol{\theta}}) + L_d(\mathbf{c}_{\boldsymbol{\phi}})$, where $\mathbf{f}_{\boldsymbol{\theta}} = (\mathbf{f}_1, \ldots, \mathbf{f}_M)$ and $\mathbf{c}_{\boldsymbol{\phi}} = (c_1, \ldots, c_M)$. We focus on minimizing $L_d(\mathbf{f}_{\boldsymbol{\theta}})$, since in different environments the prediction functions $\mathbf{f}_i$ can often be reused, while the domains may differ. As mentioned, we define $L_d(\mathbf{f}_{\boldsymbol{\theta}})$ simply as the sum of the description lengths of the numbers parametrizing $\mathbf{f}_{\boldsymbol{\theta}}$:

$$L_d(\mathbf{f}_{\boldsymbol{\theta}}) = \sum_j L_d(\theta_j). \tag{C1}$$

This means that $L_d(\mathbf{f}_{\boldsymbol{\theta}})$ can be significantly reduced if an irrational parameter is replaced by a simpler rational number.

If a physics experiment or neural net training produces a parameter $p = 1.999942$, it would be natural to interpret this as a hint, and to check if $p = 2$ gives an equally acceptable fit to the data. We formalize this by replacing any real-valued parameter p_i in our theory $\mathcal{T}$ by its nearest integer if this reduces the total description length in Eq. (4), as detailed below. We start this search for integer candidates with the parameter that is closest to an integer, refitting for the other parameters after each successful "integer snap."

What if we instead observe a parameter $p = 1.5000017$? Whereas generic real numbers have a closest integer, they lack a closest rational number. Moreover, as illustrated in Fig. 2, we care not only about closeness [to avoid increasing the second term in Eq. (4)] but also about simplicity (to reduce the first term). To rapidly find the best "rational snap" candidates (dots in Fig. 2 that lie both near p and far down), we perform a continued fraction expansion of p and use each series truncation as a rational candidate. We repeat this for all parameters in the theory $\mathcal{T}$, again accepting only those snaps that reduce the total description length. We again wish to try the most promising snap candidates first; to rapidly identify promising candidates without having to recompute the second term in Eq. (4), we evaluate all truncations of all parameters as in Fig. 3, comparing the description length of the rational approximation $q = m/n$ with the description length of the approximation error $|p - q|$. The most promising candidate minimizes their sum, i.e., lies farthest down to the left of the diagonal in the figure. The figure illustrates how, given the parameter vector $\mathbf{p} = \{\pi, \sqrt{2}, 3.43180632382353\}$, the first snap to be attempted will replace the third parameter by 53/17.

We propose Algorithm 3 to implement the above minimization of $L_d(\mathbf{f}_{\boldsymbol{\theta}})$ without increasing $L_d(\mathcal{T}, D)$ [Eq. (4)]. For each theory $\mathcal{T}_i = (\mathbf{f}_i, c_i)$, we first extract the examples $D^{(i)}$ inside its domain, then perform a series of tentative transformations (simplifications) of the prediction function $\mathbf{f}_i$ using the MinimizeDL subroutine. This subroutine takes $\mathbf{f}_i$, the transformation, and $D^{(i)}$ as inputs and repeatedly applies the transformation to $\mathbf{f}_i$. After each such transformation, it fine-tunes the fit of $\mathbf{f}_i$ to $D^{(i)}$ using gradient descent. For determining whether to accept the transformation, Algorithm 3 presents the simplest 0-step patience implementation: if the description length $\mathrm{dl} = L_d(\mathbf{f}_i) + \sum_{(\mathbf{x}_t, \mathbf{y}_t) \in D^{(i)}} \ell_{L_d, \epsilon}(\mathbf{f}_i(\mathbf{x}_t), \mathbf{y}_t)$ for theory i decreases, then apply the transformation again if possible; otherwise exit the loop. In general, to allow for temporary increase of L_d during the transformations, a nonzero patience can be adopted: at each step, save the best-performing model as the pivot model, and if L_d does not decrease during n consecutive transformations inside MinimizeDL, exit the loop. In our implementation, we use a 4-step patience.

We now detail the five transformations used in Algorithm 3. The collapseLayer transformation finds all successive layers of a neural net where the lower layer has linear activation, and combines them into one. The toSymbolic transformation transforms $\mathbf{f}_i$ from the form of a neural net into a symbolic expression (in our implementation, from a PyTorch net to a SymPy symbolic lambda expression). These two transformations are one-time transformations (for example, once $\mathbf{f}_i$ has been transformed to a symbolic expression, toSymbolic cannot be applied to it again). The

localSnap transformation successively sets the incoming weights in the first layer to 0, thus favoring inputs that are closer to the current time step. The integerSnap transformation finds the (nonsnapped) parameter in $\mathbf{f}_i$ that is closest to an integer, and snaps it to that integer. The rationalSnap transformation finds the (nonsnapped) parameter in $\mathbf{f}_i$ that has the lowest bit sum when replaced by a rational number, as described in Sec. II D, and snaps it to that rational number. The latter three transformations can be applied multiple times to $\mathbf{f}_i$, until there are no more parameters to snap in $\mathbf{f}_i$, or the transformation followed by fine-tuning fails to reduce the description length.

In the bigger picture, Algorithm 3 is an implementation of minimizing the $\mathrm{DL}(\mathbf{f}_\theta)$ without increasing the total $\mathrm{DL}(\mathcal{T}, D)$, if the description length of $\mathbf{f}_\theta$ is given by Eq. (C1). There can be other ways to encode $\mathcal{T}$ with a different formula for $\mathrm{DL}(\mathcal{T})$, in which case the transformations for decreasing $\mathrm{DL}(\mathcal{T})$ may be different. But the structure of Algorithm 3 remains the same, with the goal of minimizing $\mathrm{DL}(\mathbf{f}_\theta)$ without increasing $\mathrm{DL}(\mathcal{T}, D)$ with respect to whatever DL formula it is based on.

In the still bigger picture, Algorithm 3 is a computationally efficient approximate implementation of the MDL formalism, involving the following two approximations:

(1) The description lengths $L_d(x)$ for various types of numbers are approximate, for convenience. For example, the length of the shortest self-terminating bit string encoding an arbitrary natural number n grows slightly faster than our approximation $\log_2 n$, because self-termination requires storing not only the binary digits of the integer, but also the length of said bit string, recursively, requiring $\log_2 n + \log_2 \log_2 n + \log_2 \log_2 n + \cdots$, where only the positive terms are included [51]. Slight additional overhead is required to upgrade the encodings to actual programs in some suitable language, including encoding of whether bits encode integers, rational numbers, floating-point numbers, etc.

(2) If the above-mentioned $L_d(x)$ formulas were made exact, they would be mere *upper bounds* on the true minimum description length. For example, our algorithm gives a gigabyte description length for $\sqrt{2}$ with precision $\epsilon = 256^{-10^9}$, even though it can be computed by a rather short program, and there is no simple algorithm for determining which numbers can be accurately approximated by algebraic numbers. Computing the true minimum description length is a famous numerically intractable problem.

APPENDIX D: UNIFICATION ALGORITHM

The unification process takes as input the symbolic prediction functions $\{(\mathbf{f}_i, \cdot)\}$, and outputs master theories $\mathcal{T} = \{(\mathbf{f}_\mathbf{p}, \cdot)\}$ such that by varying each $\mathbf{p}$ in $\mathbf{f}_\mathbf{p}$, we can generate a continuum of prediction functions $\mathbf{f}_i$ within a certain class of prediction functions. The symbolic expression consists of 3 building blocks: operators (e.g., $+$, $-$, $\times$, $/$), input variables (e.g., x_1, x_2), and coefficients that can be either a rational number or irrational number. The unification algorithm first calculates the description length $\mathrm{dl}^{(i)}$ of each prediction function, then clusters them into K clusters using, e.g., K-means clustering. Within each cluster S_k, it first canonicalizes each $\mathbf{f}_{i_k} \in S_k$ into a 2-tuple $(\mathbf{g}_{i_k}, \mathbf{h}_{i_k})$, where $\mathbf{g}_{i_k}$ is a tree-form

---

Algorithm 4. AI physicist: Theory unification.

Require Hub: theory hub
Require C: initial number of clusters
1: **for** $(\mathbf{f}_i, c_i)$ **in Hub**.all-symbolic-theories **do**:
2:     $\mathrm{dl}^{(i)} \leftarrow L_d(\mathbf{f}_i)$
3: **end for**
4: $\{S_k\} \leftarrow$ Cluster $\{\mathbf{f}_i\}$ into C clusters based on $\mathrm{dl}^{(i)}$
5: **for** S_k in $\{S_k\}$ **do**:
6:     $(\mathbf{g}_{i_k}, \mathbf{h}_{i_k}) \leftarrow$ **Canonicalize**$(\mathbf{f}_{i_k})$, $\forall \mathbf{f}_{i_k} \in S_k$
7:     $\mathbf{h}_k^* \leftarrow$ Mode of $\{\mathbf{h}_{i_k} | \mathbf{f}_{i_k} \in S_k\}$.
8:     $G_k \leftarrow \{\mathbf{g}_{i_k} | \mathbf{h}_{i_k} = \mathbf{h}_k^*\}$
9:     $\mathbf{g}_{\mathbf{p}_k} \leftarrow$ Traverse all $\mathbf{g}_{i_k} \in G_k$ with synchronized steps, replacing the coefficient by a $\mathbf{p}_{j_k}$ when not all coefficients at the same position are identical.
10:     $\mathbf{f}_{\mathbf{p}_k} \leftarrow$ toPlainForm$(\mathbf{g}_{\mathbf{p}_k})$
11: **end for**
12: $\mathcal{T} \leftarrow \{(\mathbf{f}_{\mathbf{p}_k}, \cdot)\}, k = 1, 2, \ldots, C$
13: $\mathcal{T} \leftarrow$ MergeSameForm$(\mathcal{T})$
14: **return** $\mathcal{T}$

---

subroutine Canonicalize$(\mathbf{f}_i)$:
s1: $\mathbf{g}_i \leftarrow$ ToTreeForm$(\mathbf{f}_i)$
s2: $\mathbf{h}_i \leftarrow$ Replace all non-input coefficients by a symbol s
return $(\mathbf{g}_i, \mathbf{h}_i)$

---

expression of $\mathbf{f}_{i_k}$ where each internal node is an operator, and each leaf is an input variable or a coefficient. When multiple orderings are equivalent (e.g., $x_1 + x_2 + x_3$ vs $x_1 + x_3 + x_2$), it always uses a predefined partial ordering. $\mathbf{h}_{i_k}$ is the structure of $\mathbf{g}_{i_k}$ where all coefficients are replaced by an s symbol. Then the algorithm obtains a set of $\mathbf{g}_{i_k}$ that has the same structure $\mathbf{h}_{i_k}$ with the largest cardinality (steps 7–8). This will eliminate some expressions within the cluster that might interfere with the following unification process. Step 9 is the core part, where it traverses each $\mathbf{g}_{i_k} \in G_k$ with synchronized steps using, e.g., depth-first search or breath-first search. This is possible since each $\mathbf{g}_{i_k} \in G_k$ has the same tree structure h_k^*. During traversing, whenever encountering a coefficient and not all coefficients across G_k at this position are the same, replace the coefficients by some symbol $\mathbf{p}_{j_k}$ that has not been used before. Essentially, we are turning all coefficients that vary across G_k into a parameter, and the coefficients that do not vary stay as they are. In this way, we obtain a master prediction function $\mathbf{f}_{\mathbf{p}_k}$. Finally, at step 13, the algorithm merges the master prediction functions in $\mathcal{T} = \{(\mathbf{f}_{\mathbf{p}_k}, \cdot)\}$ that have the exact same form, and return $\mathcal{T}$. The domain classifier is neglected during the unification process, since at different environments, each prediction function can have vastly different spatial domains. It is the prediction function (which characterizes the equation of motion) that is important for generalization.

APPENDIX E: ADDING AND PROPOSING THEORIES

Here we detail the algorithms adding theories to the hub and proposing them for use in new environments. Algorithm 5 provides a simplest version of the theory-proposing algorithm. Given a new data set D, the theory hub inspects all theories i,

Algorithm 5. AI physicist: Theory proposing from hub.

Require Hub: theory hub
Require Data set $D = \{(\mathbf{x}_t, \mathbf{y}_t)\}$
Require M_0: number of theories to propose from the hub
1: $\{(\mathbf{f}_i, c_i)\} \leftarrow \textbf{Hub}.\text{retrieve-all-theories}()$
2: $D_{\text{best}}^{(i)} \leftarrow \{(\mathbf{x}_t, \mathbf{y}_t) | \arg\min_j \ell_{L_d, \epsilon}[\mathbf{f}_j(\mathbf{x}_t), \mathbf{y}_t] = i\}, \forall i$
3: $\mathcal{T}_{M_0} \leftarrow \{(\mathbf{f}_i, c_i) | D_{\text{best}}^{(i)} \text{ ranks among } M_0 \text{ largest sets in } \{D_{\text{best}}^{(i)}\}\}$
4: **return** $\mathcal{T}_{M_0}$

---

and for each one, counts the number n_i of data points where it outperforms all other theories. The top M_0 theories with largest n_i are then proposed.

For theory adding after training with DDAC (Algorithm 2), each theory i calculates its description length $\text{dl}^{(i)}$ inside its domain. If its $\text{dl}^{(i)}$ is smaller than a threshold η, then the theory $(\mathbf{f}_i, c_i)$ with its corresponding examples $D^{(i)}$ are added to the theory hub. The reason why the data $D^{(i)}$ are also added to the hub is that $D^{(i)}$ gives a reference for how the theory $(\mathbf{f}_i, c_i)$ was trained, and is also needed in the Occam's razor algorithm.

APPENDIX F: TIME COMPLEXITY

In this Appendix, we list crude estimates of the time complexity of our AI physicist algorithm, i.e., of how its runtime scales with key parameters.

DDAC, the differentiable divide-and-conquer algorithm (Algorithm 2), is run once for each of the n_{myst} different mystery worlds, with a total runtime scaling roughly as

$$O\big(n_{\text{myst}} n_{\text{par}} n_{\text{data}} n_{\text{dom}}^2\big),$$

where n_{par} is the average number of neural-network parameters in a theory, n_{data} is the average number of data points (time steps) per mystery, and n_{dom} is the number of discovered domains (in our case $\leqslant 4$). The power of two for n_{dom} appears because the time to evaluate the loss function scales as n_{dom}, and we need to perform of order n_{dom} training cycles to add the right number of theories. The n_{par} scaling is due to that the forward and backward propagation of the neural net involves successive matrices multiplied by a vector, which scales as $O(n^2)$, where $n \simeq \sqrt{n_{\text{par}}/N_{\text{lay}}^f}$ is the matrix dimension for each

Algorithm 6. AI physicist: Adding theories to hub.

Require Hub: theory hub
Require $\mathcal{T} = \{(\mathbf{f}_i, c_i)\}$: Trained theories from Alg. 2
Require Data set $D = \{(\mathbf{x}_t, \mathbf{y}_t)\}$
Require η: L_d threshold for adding theories to hub
1: $D^{(i)} \leftarrow \{(\mathbf{x}_t, \mathbf{y}_t) | \arg\max_j \{c_j(\mathbf{x}_t)\} = i\}, \forall i$
2: $\text{dl}^{(i)} \leftarrow \frac{1}{|D^{(i)}|} \sum_{(\mathbf{x}_t, \mathbf{y}_t) \in D^{(i)}} \ell_{L_d, \epsilon}(\mathbf{f}_i(\mathbf{x}_t), \mathbf{y}_t), \forall i$
3: **for** i in $\{1, 2, \ldots, |\mathcal{T}|\}$ **do**:
4:      **if** $\text{dl}^{(i)} < \eta$ **do**
5:          **Hub**.addIndividualTheory$((\mathbf{f}_i, c_i), D^{(i)})$
6:      **end if**
7: **end for**

---

layer and N_{lay}^f is the number of layers. Accumulating all layers we have $N_{\text{lay}}^f n^2 = n_{\text{par}}$. We make no attempt to model how the number of training epochs needed to attain the desired accuracy depends on parameters.

Our *lifelong learning* algorithm is also run once per mystery, with a time cost dominated by that for proposing new theories (Algorithm 5), which scales as

$$O(n_{\text{myst}} n_{\text{data}} n_{\text{theo}}).$$

Here n_{theo} is the number of theories in the theory hub.

In contrast, our *Occam's razor* algorithm (Algorithm 3) and *unification* algorithm (Algorithm 4) are run once per learned theory, not once per mystery. For Occam's razor, the total runtime is dominated by that for snapping to rational numbers, which scales as

$$O(n_{\text{par}} n_{\text{data}} n_{\text{theo}}).$$

For the unification, the total runtime scales as $O(n_{\text{par}} n_{\text{theo}})$, which can be neglected relative to the cost of Occam's razor.

We note that all these algorithms have merely polynomial time complexity. The DDAC algorithm dominates the time cost; our mystery worlds were typically solved in about 1 hour each on a single CPU. If vast amounts of data are available, it may suffice to analyze a random subset of much smaller size.

APPENDIX G: PROOF OF THEOREM 1 AND COROLLARY

Here we give the proof for Theorem 1, restated here for convenience.

Theorem 1. Let $\hat{\mathbf{y}}_t^{(i)} \equiv \mathbf{f}_i(\mathbf{x}_t)$ denote the prediction of the target $\mathbf{y}_t$ by the function $\mathbf{f}_i$, $i = 1, 2, \ldots, M$. Suppose that $\gamma < 0$ and $\ell(\hat{\mathbf{y}}_t, \mathbf{y}_t) = \ell(|\hat{\mathbf{y}}_t - \mathbf{y}_t|)$ for a monotonically increasing function $\ell(u)$ that vanishes on $[0, u_0]$ for some $u_0 \geqslant 0$, with $\ell(u)^\gamma$ differentiable and strictly convex for $u > u_0$. Then if $0 < \ell(\hat{\mathbf{y}}_t^{(i)}, \mathbf{y}_t) < \ell(\hat{\mathbf{y}}_t^{(j)}, \mathbf{y}_t)$, we have

$$\left| \frac{\partial \mathcal{L}_\gamma}{\partial u_t^{(i)}} \right| > \left| \frac{\partial \mathcal{L}_\gamma}{\partial u_t^{(j)}} \right|, \tag{G1}$$

where $u_t^{(i)} \equiv |\hat{\mathbf{y}}_t^{(i)} - \mathbf{y}_t|$.

Proof. Since $u_t^{(i)} \equiv |\hat{\mathbf{y}}_t^{(i)} - \mathbf{y}_t|$ and $\ell(\hat{\mathbf{y}}_t, \mathbf{y}_t) = \ell(|\hat{\mathbf{y}}_t - \mathbf{y}_t|)$, the generalized mean loss L_γ as defined in Eq. (3) can be rewritten as

$$\mathcal{L}_\gamma = \sum_t \left[\frac{1}{M} \sum_{k=1}^M \ell(u_t^{(k)})^\gamma \right]^{\frac{1}{\gamma}}, \tag{G2}$$

which implies that

$$\left| \frac{\partial \mathcal{L}_\gamma}{\partial u_t^{(i)}} \right| = \left| \frac{1}{\gamma M} \left[\frac{1}{M} \sum_{k=1}^M \ell(u_t^{(k)})^\gamma \right]^{\frac{1}{\gamma} - 1} \frac{d\ell(u_t^{(i)})^\gamma}{du_t^{(i)}} \right|$$

$$= \frac{1}{|\gamma| M} \left[\frac{1}{M} \sum_{k=1}^M \ell(u_t^{(k)})^\gamma \right]^{\frac{1}{\gamma} - 1} \left| \frac{d\ell(u_t^{(i)})^\gamma}{du_t^{(i)}} \right|.$$

Since only the last factor depends on i, proving Eq. (G1) is equivalent to proving that

$$\left| \frac{d\ell(u_t^{(i)})^\gamma}{du_t^{(i)}} \right| > \left| \frac{d\ell(u_t^{(j)})^\gamma}{du_t^{(j)}} \right|. \tag{G3}$$

Let us henceforth consider only the case $u > u_0$, since the conditions $\ell(u_t^{(j)}) > \ell(u_t^{(i)}) > 0$ imply $u_t^{(j)} > u_t^{(i)} > u_0$. Since $\gamma < 0$, $\ell(u) > 0$, and $\ell'(u) \geqslant 0$, we have $\frac{d\ell(u)^\gamma}{du} = \gamma\ell(u)^{\gamma-1}\ell'(u) \leqslant 0$, so that $\left|\frac{d\ell(u)^\gamma}{du}\right| = -\frac{d\ell(u)^\gamma}{du}$. Because $\ell(u)^\gamma$ is differentiable and strictly convex, its derivative $\frac{d\ell(u)^\gamma}{du}$ is monotonically increasing, implying that $\left|\frac{d\ell(u)^\gamma}{du}\right| = -\frac{d\ell(u)^\gamma}{du}$ is monotonically decreasing. Thus $\left|\frac{d\ell(u_1)^\gamma}{du_1}\right| > \left|\frac{d\ell(u_2)^\gamma}{du_2}\right|$ whenever $u_1 < u_2$. Setting $u_1 = |\hat{\mathbf{y}}_t^{(i)} - \mathbf{y}_t|$ and $u_2 = |\hat{\mathbf{y}}_t^{(j)} - \mathbf{y}_t|$ therefore implies Eq. (G3), which completes the proof. ∎

The following Corollary 1 demonstrates that the theorem applies to several popular loss functions as well as our two description-length loss functions.

Corollary 1. Defining $u \equiv |\hat{\mathbf{y}} - \mathbf{y}|$, the following loss functions which depend only on u satisfy the conditions for Theorem 1:

(1) $\ell(u) = u^r$ for any $r > 0$, which includes MSE loss ($r = 2$) and mean-absolute-error loss ($r = 1$).

(2) Huber loss:

$$\ell_\delta(u) = \begin{cases} \frac{1}{2}u^2, & u \in [0, \delta], \\ \delta(u - \frac{\delta}{2}), & \text{otherwise,} \end{cases}$$

where $\delta > 0$.

(3) Description length loss

$$\ell_{L_d,\epsilon}(u) = \frac{1}{2}\log_2\left[1 + \left(\frac{u}{\epsilon}\right)^2\right].$$

(4) Hard description length loss

$$\ell_{\text{DLhard},\epsilon}(u) = \log_2 \max\left(1, \frac{u}{\epsilon}\right).$$

Proof. We have $u_0 = 0$ for (1), (2), (3), and $u_0 = \epsilon$ for (4). All four functions ℓ are monotonically increasing, satisfy $\ell(0) = 0$, and are differentiable for $u > u_0$, so all that remains to be shown is that $\ell(u)^\gamma$ is strictly convex for $u > u_0$, i.e., that $\frac{d^2\ell(u)^\gamma}{du^2} > 0$ when $u > u_0$.

(1) For $\ell(u) = u^r$ and $u > 0$, we have $\frac{d^2\ell(u)^\gamma}{du^2} = \gamma r(\gamma r - 1)u^{\gamma r - 2} > 0$, since $\gamma < 0$ and $r > 0$ implies that $\gamma r < 0$ and $\gamma r - 1 < 0$, so $\ell(u)^\gamma$ is strictly convex for $u > 0$.

(2) The Huber loss $\ell_\delta(u)$ is continuous with a continuous derivative. It satisfies $\frac{d^2\ell(u)^\gamma}{du^2} > 0$ both for $0 < u < \delta$ and for $\delta < u$ according to the above proof of (1), since $\ell_\delta(u)$ is proportional to ℓ^r in these two intervals with $r = 2$ and $r = 1$, respectively. At the transition point $u = \delta$, this second derivative is discontinuous, but takes positive value approaching both from the left and from the right, so $\ell(u)^\gamma$ is strictly convex. More generally, any function $\ell(u)$ built by smoothly connecting functions $\ell_i(u)$ in different intervals will satisfy our theorem if the functions $\ell_i(u)$ do.

(3) Proving strict convexity of $\ell(u)^\gamma$ when ℓ is the description-length loss $\ell_{L_d,\epsilon}(u) = \frac{1}{2}\log_2[1 + (\frac{u}{\epsilon})^2]$ is equivalent to

proving it when $\ell(u) = \rho(u) \equiv \ln(1 + u^2)$, since convexity is invariant under horizontal and vertical scaling. We thus need to prove that

$$\frac{d^2\rho(u)^\gamma}{du^2} = -\frac{2\gamma[\ln(1 + u^2)]^{\gamma-2}}{(1 + u^2)^2[2u^2(1 - \gamma) + (u^2 - 1)\ln(1 + u^2)]}$$

is positive when $u > 0$. The factor $\frac{-2\gamma[\ln(1+u^2)]^{\gamma-2}}{(1+u^2)^2}$ is always positive. The other factor

$$2u^2(1 - \gamma) + (u^2 - 1)\log(1 + u^2) > 2u^2$$
$$+ (u^2 - 1)\log(1 + u^2),$$

since $\gamma < 0$. Now we only have to prove that the function

$$\chi(u) \equiv 2u^2 + (u^2 - 1)\log(1 + u^2) > 0$$

when $u > 0$. We have $\chi(0) = 0$ and

$$\chi'(u) = 2u\left[\frac{1 + 3u^2}{1 + u^2} + \log(1 + u^2)\right] > 0$$

when $u > 0$. Therefore $\chi(u) = \chi(0) + \int_0^u \chi'(u')du' > 0$ when $u > 0$, which completes the proof that $\ell_{L_d,\epsilon}(u)^\gamma$ is strictly convex for $u > 0$.

(4) For the hard description length loss $\ell_{\text{DLhard},\epsilon}(u) = \log_2 \max(1, \frac{u}{\epsilon})$, we have $u_0 = \epsilon$. When $u > \epsilon$, we have $\ell'_{\text{DLhard},\epsilon}(u) = \frac{1}{u\ln 2} > 0$ and

$$\frac{d^2\ell_{\text{DLhard},\epsilon}^\gamma(u)}{du^2} = \frac{\gamma}{\ln 2}\left(-1 + \gamma - \ln\frac{u}{\epsilon}\right)\frac{\left(\ln\frac{u}{\epsilon}\right)^{\gamma-2}}{u^2}.$$

For $u > \epsilon$, the factor $\frac{(\ln\frac{u}{\epsilon})^{\gamma-2}}{u^2}$ is always positive, as is the factor $\gamma(-1 + \gamma - \ln\frac{u}{\epsilon})$, since $\gamma < 0$. $\ell_{\text{DLhard},\epsilon}^\gamma(u)$ is therefore strictly convex for $u > \epsilon$. ∎

APPENDIX H: ELIMINATING TRANSITION DOMAINS

In this Appendix, we show how the only hard problem our AI physicist need solve is to determine the laws of motion far from domain boundaries, because once this is done, the exact boundaries and transition regions can be determined automatically.

Our AI physicist tries to predict the next position vector $\mathbf{y}_t \in R^d$ from the concatenation $\mathbf{x}_t = (\mathbf{y}_{t-T}, \ldots, \mathbf{y}_{t-1})$ of the last T position vectors. Consider the example shown in Fig. 6, where motion is predicted from the last $T = 3$ positions in a space with $d = 2$ dimensions containing $n = 2$ domains with different physics (an electromagnetic field in the upper left quadrant and free motion elsewhere), as well as perfectly reflective boundaries. Although there are only two physics domains in the 2-dimensional space, there are many more types of domains in the $Td = 6$-dimensional space of $\mathbf{x}_t$ from which the AI physicist makes its predictions of $\mathbf{y}_t$. When a trajectory crosses the boundary between the two spatial regions, there can be instances where $\mathbf{x}_t$ contains 3, 2, 1, or 0 points in the first domain and correspondingly 0, 1, 2, or 3 points in the second domain. Similarly, when the ball bounces, there can be instances where $\mathbf{x}_t$ contains 3, 2, 1, or 0 points before the bounce and correspondingly 0, 1, 2, or 3 points after. Each of these situations involves a different function $\mathbf{x}_t \mapsto \mathbf{y}_t$ and a corresponding 6-dimensional domain of validity for the AI physicist to learn.
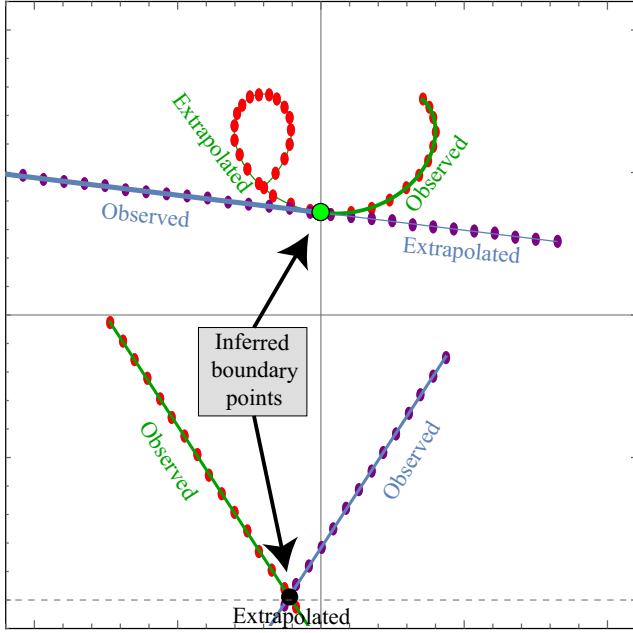
FIG. 6. Points where forward and backward extrapolations agree (large black dots) are boundary points. The tangent vectors agree for region boundaries (upper example), but not for bounce boundaries (lower example).
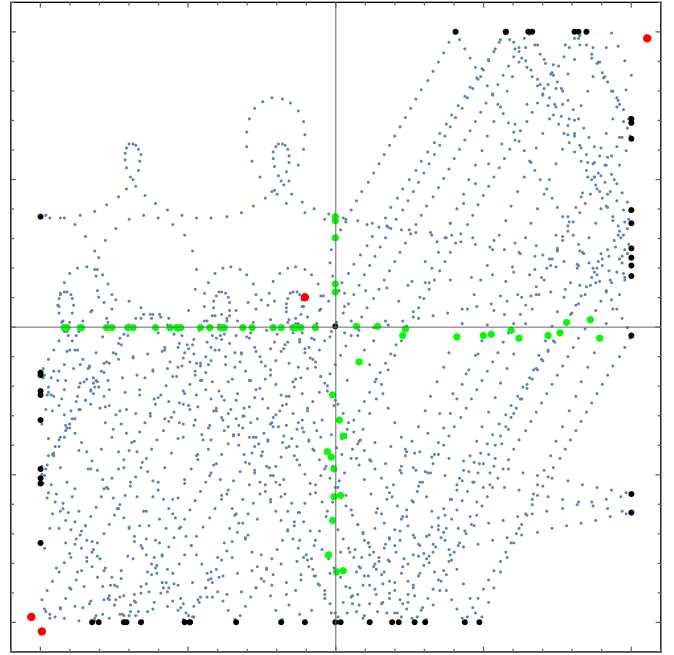

FIG. 7. Example of automatically determined boundary points, for region boundary points (green), bounce boundary points (black), and failed cases (red).

Our numerical experiments showed that the AI physicist typically solves the big domains (where all vectors in $\mathbf{x}_t$ lie in the same spatial region), but occasionally fails to find an accurate solution in some of the many small transition domains involving boundary crossings or bounces, where data are insufficient. Fortunately, simple postprocessing can automatically eliminate these annoying transition domains with an algorithm that we will now describe.

The first step of the algorithm is illustrated in Fig. 6. For each big domain where our AI physicist has discovered the future-predicting function $\mathbf{x}_t \mapsto \mathbf{y}_t$, we determine the corresponding function that predicts the *past* ($\mathbf{x}_t \mapsto \mathbf{y}_{t-T-1}$) by fitting to forward trajectories generated with random initial conditions. Now whenever a trajectory passes from a big domain through a transition region into another big domain, two different extrapolations can be performed: forward in time from the first big domain or backward in time from the second big domain. Using cubic spline interpolation, we fit continuous functions $\mathbf{y}_f(t)$ and $\mathbf{y}_b(t)$ (smooth curves in Fig. 6) to these forward-extrapolated and backward-extrapolated trajectories, and numerically find the time

$$t_* \equiv \arg\min |\mathbf{y}_f(t) - \mathbf{y}_b(t)| \tag{H1}$$

when the distance between the two predicted ball positions is minimized. If this minimum is numerically consistent with zero, so that $\mathbf{y}_f(t_*) \approx \mathbf{y}_b(t_*)$, then we record this as being a boundary point. If both extrapolations have the same derivative there, i.e., if $\mathbf{y}_f'(t_*) \approx \mathbf{y}_b'(t_*)$, then it is an interior boundary point between two different regions (Fig. 6, top); otherwise it is an external boundary point where the ball bounces (Fig. 6, bottom).

Figure 7 shows these two types of automatically computed boundary points in green and black, respectively. These can now be used to retrain the domain classifiers to extend the big domains to their full extent, eliminating the transition regions.

Occasionally the boundary point determinations will fail because of multiple transitions within T time steps; Fig. 7 illustrates that these failures (red dots) force us to discard merely a tiny fraction of all cases, thus having a negligible effect on the ability to fit for the domain boundaries.

APPENDIX I: NUMERICAL EXPERIMENT DETAILS

In this Appendix, we provide supplementary details on our benchmark problems.

1. Mystery worlds

World generation. Our mystery worlds consist of a ball elastically bouncing against the square boundary of the two-dimensional spatial region where $|x| \leqslant 2$ and $|y| \leqslant 2$ (see Fig. 4). In each of the four quadrants, one of the following laws of physics is selected, together with the parameters sampled from distributions as follows:

(1) Free motion.

(2) A uniform gravitational field $\mathbf{g} = (g_x, g_y, 0)$ with g_x, g_y drawn from a uniform distribution: $g_x, g_y \sim U[-5, 5]$.

(3) Harmonic motion with frequency ω around a line a distance a from the origin, making an angle ϕ with the x axis; $\omega \sim U[1, 4]$, $a \sim U[0.2, 0.5]$, $\phi \sim U[0, 2\pi]$.

(4) A uniform electric field $\mathbf{E} = (E_x, E_y, 0)$ and magnetic field $\mathbf{B} = (0, 0, B_z)$; $E_x, E_y \sim U[-5, 5]$, $B_z \sim U[0, 10]$.

To control the difficulty of the tasks and avoid near-degenerate scenarios, we keep only mystery worlds satisfying the following two criteria: (1) At least 0.01 separation between all equations of motion (EOMs) in the same world, defined as the Euclidean distance between the vectors of coefficients specifying the EOM difference equations, and (2) at least 0.0015 of any noninteger parameter from its nearest integer.

Trajectory simulation. Within each world, we initialize the ball with a random position $(x, y) \sim U[-1, 1]^2$ and velocity $(v_0 \cos \theta_0, v_0 \sin \theta_0, 0)$; $v_0 \sim U[0.1, 0.5]$, $\theta_0 \sim U[0, 2\pi]$. We then compute its position for $N = 4\,000$ times steps $t = 1, 2, \ldots, N$ with time interval 0.05.

Although the above-mentioned laws of physics are linear, the mapping from past points $(\mathbf{y}_{t-T}, \ldots, \mathbf{y}_{t-1})$ to the next points $\mathbf{y}_t$ is generally nonlinear because of region boundaries where the ball either bounces or transitions to a different physics region. An exception is when three successive points lie within the same region (with the same physics), which happens far from boundaries: in this case, the mapping from $(\mathbf{y}_{t-2}, \mathbf{y}_{t-1}) \mapsto \mathbf{y}_t$ is deterministic and linear thanks to the differential equations of motion being second order and linear.

Architecture. For the newborn and AI physicist agents, each prediction function $\mathbf{f}_i$ is implemented as an N^f_{lay}-layer neural network with linear activation, with N^f_{neur}-neuron hidden layers (we use $N^f_{\text{lay}} = 3$, $N^f_{\text{neur}} = 8$ for our main experiments; see Table V). Each domain subclassifier c_i is implemented as an N^c_{lay}-layer neural net, with two hidden N^c_{neur}-neuron layers with leakyReLU activation $\sigma(x) = \max\{0.3x, x\}$, and the output layer having linear activation (we use $N^c_{\text{lay}} = 3$, $N^c_{\text{neur}} = 8$ for our main experiments). The baseline model is implemented as a single N^f_{lay}-layer neural net with two hidden 16-neuron layers with leakyReLU activation followed by a linear output layer. Note that for a fair comparison, the baseline model has more hidden neurons, to roughly compensate for the newborn and AI physicist agents typically having multiple theories. The baseline network is nonlinear to boost its expressive power for modeling the nonlinear prediction function of each world as a whole. For the domain classifier $\mathbf{c} = (c_1, c_2, \ldots, c_M)$, it is an N^c_{lay}-layer neural net where each hidden layer has $N^c_{\text{neur}} = 8$ neurons and leakyReLU activation. The last layer has linear activation. See Table V for a list of hyperparameters.

Evaluation. The unsupervised classification accuracy is defined as the fraction of correctly classified points, using the permutation of the learned domain labels that best matches the hidden ground truth domain labels. It is "unsupervised" in the sense that there is no external supervision signal as to which domain label each point should be assigned to: the AI physicist has to figure out the number of domains and their boundaries and assign each point to a domain, which is a difficult task.

We define a domain as *solved* if the agent discovers its law of motion as a difference equation (prediction function) within the following stringent tolerance: all rational coefficients in the difference equation are exactly matched, and all irrational coefficients agree to an accuracy better than 10^{-4}. Because of the nature of the physics problems, some of these difference equation coefficients take on the values $0, -1,$ or 2, so solving a region requires successful integer snapping as described in Sec. II D. To make the problem even harder, we also fine-tune the magnetic field in five of the electromagnetic regions to make some of the coefficients simple fractions such as $1/3$ and $1/4$, thus making solving those regions contingent on successful rational snapping as described in Sec. II D. Domain solving can fail either by "undersnapping" (failing to approximate a floating-point number by a rational number) or "oversnapping" (mistakenly rounding to a rational number). All our mystery worlds are available for download [66].

As shown in Appendix H, the only hard problem our AI physicist or other algorithms need to solve is to determine the laws of motion away from domain boundaries. Therefore, we evaluate, tabulate, and compare the performance of the algorithms only on interior points, i.e., excluding data points $(\mathbf{x}_t, \mathbf{y}_t)$ straddling a boundary encounter.

2. Double pendulum

Our double pendulum is implemented as two connected pendulums with massless rods of length 1 and that each have a point charge of 1 at their end. As illustrated in Fig. 5, the system state is fully determined by the 4-tuple $\mathbf{y} = (\theta_1, \dot{\theta}_1, \theta_2, \dot{\theta}_2)$ and immersed in a piecewise constant electric field $\mathbf{E}$: $\mathbf{E} = (0, -E_1)$ in the upper half plane $y \geqslant -1.05$, and $\mathbf{E} = (0, E_2)$ in the lower half plane $y < -1.05$, using coordinates where y increases vertically and the origin is at the pivot point of the upper rod.

We generate 7 environments by setting (E_1, E_2) equal to $(E_0, 2E_0)$, $(E_0, 1.5E_0)$, (E_0, E_0), $(E_0, 0.5E_0)$, $(2E_0, E_0)$, $(1.5E_0, E_0)$, and $(0.5E_0, E_0)$, where $E_0 = 9.8$. We see that there are two different EOMs for the double-pendulum system depending on which of the two fields the lower charge is in (the upper charge is always in E_1). We use Runge-Kutta numerical integration to simulate $\mathbf{y} = (\theta_1, \dot{\theta}_1, \theta_2, \dot{\theta}_2)$ for 10 000 time steps with interval of 0.05, and the algorithms' task is to predict the future $(\mathbf{y}_{t+1})$ based on the past ($\mathbf{x}_t \equiv \mathbf{y}_t$; history length $T = 1$), and simultaneously discover the two domains and their different EOMs unsupervised.

In this experiment, we implement the prediction function of the baseline and newborn both as an N^f_{lay}-layer neural net (we use $N^f_{\text{lay}} = 6$) during DDAC. For the newborn, each hidden layer has $N^f_{\text{neur}} = 160$ neurons with hyperbolic tangent (tanh) activation, and for the baseline, each hidden layer has $N^f_{\text{neur}} = 320$ neurons with tanh activation for a fair comparison. For the newborn, the optional AddTheories($\mathcal{T}, D$) (step s8 in Algorithm 2) is turned off to prevent unlimited adding of theories. The initial number M of theories for the newborn is set to $M = 2$ and $M = 3$, each run with 10 instances with random initialization. Its domain classifier $\mathbf{c} = (c_1, c_2, \ldots, c_M)$ is an N^c_{lay}-layer neural net (we use $N^c_{\text{lay}} = 3$) where each hidden layer has $N^c_{\text{neur}} = 6$ neurons and leakyReLU activation. The last layer has linear activation.

[1] Y. LeCun, Y. Bengio, and G. Hinton, Nature (London) **521**, 436 (2015).

[2] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner *et al.*, arXiv:1806.01261.

[3] S. Russell, D. Dewey, and M. Tegmark, AI Magazine **36**, 105 (2015).

[4] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, arXiv:1606.06565.

[5] M. Boden, J. Bryson, D. Caldwell, K. Dautenhahn, L. Edwards, S. Kember, P. Newman, V. Parry, G. Pegman, T. Rodden *et al.*, Connection Sci. **29**, 124 (2017).

[6] V. Krakovna and F. Doshi-Velez, arXiv:1606.05320.

[7] V. C. Müller and N. Bostrom, *Fundamental Issues of Artificial Intelligence* (Springer, Berlin, 2016), pp. 555–572.

[8] K. Grace, J. Salvatier, A. Dafoe, B. Zhang, and O. Evans, J. Artif. Intell. Res. **62**, 729 (2018).

[9] A. Graves, G. Wayne, and I. Danihelka, arXiv:1410.5401.

[10] S. Sukhbaatar, A. Szlam, J. Weston, and R. Fergus, in *Advances in Neural Information Processing Systems 28*, edited by C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett (Curran Associates, Inc., 2015), pp. 2440–2448.

[11] S. Reed and N. De Freitas, arXiv:1511.06279.

[12] E. Parisotto, A.-R. Mohamed, R. Singh, L. Li, D. Zhou, and P. Kohli, arXiv:1611.01855.

[13] J. Devlin, J. Uesato, S. Bhupatiraju, R. Singh, A.-R. Mohamed, and P. Kohli, arXiv:1703.07469.

[14] N. Bramley, E. Schulz, F. Xu, and J. Tenenbaum (unpublished).

[15] S. Muggleton, New Generation Comp. **8**, 295 (1991).

[16] N. Lavrac and S. Dzeroski, *Workshop on Logic Programming (WLP)* (Springer, Berlin, 1994), pp. 146–160.

[17] P. Liang, M. I. Jordan, and D. Klein, in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)* (International Machine Learning Society, 2010), pp. 639–646.

[18] K. Ellis, A. Solar-Lezama, and J. Tenenbaum, in *Advances in Neural Information Processing Systems 28* (Ref. [10]), pp. 973–981.

[19] E. Dechter, J. Malmaud, R. P. Adams, and J. B. Tenenbaum, in *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI-13)* (AAAI Press, 2013), pp. 1302–1309.

[20] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms* (MIT Press, 2009).

[21] J. Fürnkranz, Artif. Intell. Rev. **13**, 3 (1999).

[22] D. Ghosh, A. Singh, A. Rajeswaran, V. Kumar, and S. Levine, arXiv:1711.09874.

[23] J. Rissanen, Automatica **14**, 465 (1978).

[24] B. Hassibi and D. G. Stork, in *Advances in Neural Information Processing Systems 5 (NIPS 1992)*, edited by S. J. Hanson, J. D. Cowan, and C. L. Giles (Morgan-Kaufmann, 1993), pp. 164–171.

[25] K. Suzuki, I. Horiba, and N. Sugie, Neural Proc. Lett. **13**, 43 (2001).

[26] P. D. Grünwald, I. J. Myung, and M. A. Pitt, *Advances in Minimum Description Length: Theory and Applications* (MIT Press, 2005).

[27] S. Han, H. Mao, and W. J. Dally, arXiv:1510.00149.

[28] S. Han, J. Pool, J. Tran, and W. Dally, *Advances in Neural Information Processing Systems (NIPS 2015)* (MIT Press, 2015), pp. 1135–1143.

[29] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska *et al.*, Proc. Natl. Acad. Sci. USA **114**, 3521 (2017).

[30] Z. Li and D. Hoiem, IEEE Trans. Pattern Anal. Mach. Int. **40**, 2935 (2018).

[31] D. Lopez-Paz and M. A. Ranzato, in *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, edited by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Curran Associates, Inc., 2017), pp. 6467–6476.

[32] C. V. Nguyen, Y. Li, T. D. Bui, and R. E. Turner, arXiv:1710.10628.

[33] I. Yildirim, K. A. Smith, M. Belledonne, J. Wu, and J. B. Tenenbaum, in *2nd Conference on Cognitive Computational Neuroscience* (CCN, 2018).

[34] D. Zheng, V. Luo, J. Wu, and J. B. Tenenbaum, arXiv:1807.09244.

[35] P. Battaglia, R. Pascanu, M. Lai, D. J. Rezende, and k. kavukcuoglu, in *Advances in Neural Information Processing Systems 29 (NIPS 2016)*, edited by D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (Curran Associates, Inc., 2016), pp. 4502–4510.

[36] M. B. Chang, T. Ullman, A. Torralba, and J. B. Tenenbaum, arXiv:1612.00341.

[37] N. Watters, D. Zoran, T. Weber, P. Battaglia, R. Pascanu, and A. Tacchetti, in *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, edited by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Curran Associates, Inc., 2017), pp. 4539–4547.

[38] R. Iten, T. Metger, H. Wilming, L. Del Rio, and R. Renner, arXiv:1807.10300.

[39] B. M. Lake, T. D. Ullman, J. B. Tenenbaum, and S. J. Gershman, Behavioral Brain Sci. **40**, e253 (2017).

[40] B. C. Daniels and I. Nemenman, Nat. Commun. **6**, 8133 (2015).

[41] S. Džeroski, P. Langley, and L. Todorovski, in *Computational Discovery of Scientific Knowledge: Introduction, Techniques, and Applications in Environmental and Life Sciences*, edited by S. Džeroski and L. Todorovski (Springer, Berlin Heidelberg, 2007), pp. 1–14.

[42] P. Langley, Cognitive Science **5**, 31 (1981).

[43] P. Langley and J. M. Zytkow, Artificial Intelligence **40**, 283 (1989).

[44] S. Dzeroski and L. Todorovski, J. Intell. Inf. Syst. **4**, 89 (1995).

[45] E. Bradley, M. Easley, and R. Stolle, Artificial Intelligence **133**, 139 (2001).

[46] P. Langley, D. George, S. and Bay, and K. Saito, in *Proceedings of the Twentieth International Conference on International Conference on Machine Learning* (AAAI Press, Washington, DC, USA, 2003), pp. 432–439.

[47] P. Langley and A. Arvay, *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence* (AAAI Press, Washington, DC, USA, 2005).

[48] Y. Freund and R. E. Schapire, J. Comput. Syst. Sci. **55**, 119 (1997).

[49] R. J. Solomonoff, Inf. Cont. **7**, 1 (1964).

[50] M. Hutter, arXiv:cs/0004001.

[51] J. Rissanen, Annals Stat. **11**, 416 (1983).

[52] G. E. Hinton and D. van Camp, in *Proceedings of the Sixth Annual Conference on Computational Learning Theory* (ACM, New York, 1993), pp. 5–13.

[53] L. Blier and Y. Ollivier (unpublished).

[54] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. Le, and A. Kurakin, arXiv:1703.01041.

[55] B. Zoph and Q. V. Le, arXiv:1611.01578.

[56] B. Baker, O. Gupta, N. Naik, and R. Raskar, arXiv:1611.02167.

[57] M. Schmidt and H. Lipson, Science **324**, 81 (2009).

[58] S.-M. Udrescu and M. Tegmark, arXiv:1905.11481.

[59] J. Carrasquilla, G. Torlai, R. G. Melko, and L. Aolita, Nat. Mach. Intell. **1**, 155 (2019).

[60] J. Carrasquilla and R. G. Melko, Nat. Phys. **13**, 431 (2017).

[61] L. Wang, Phys. Rev. B **94**, 195105 (2016).

[62] E. P. Van Nieuwenburg, Y.-H. Liu, and S. D. Huber, Nat. Phys. **13**, 435 (2017).

[63] C. Lam and D. Kipping, Mon. Not. R. Astron. Soc. **476**, 5692 (2018).

[64] P. Baldi, P. Sadowski, and D. Whiteson, Nat. Commun. **5**, 4308 (2014).

[65] D. P. Kingma and J. Ba, arXiv:1412.6980.

[66] See http://space.mit.edu/home/tegmark/aiphysicist.html.