SIMULATION INTELLIGENCE: TOWARDS A NEW GENERATION OF SCIENTIFIC METHODS

Alexander Lavin[*]
Institute for Simulation Intelligence

Hector Zenil
Alan Turing Institute

Brooks Paige
Alan Turing Institute

David Krakauer
Santa Fe Institute

Justin Gottschlich
Intel Labs

Tim Mattson
Intel

Anima Anandkumar
Nvidia

Sanjay Choudry
Nvidia

Kamil Rocki
Neuralink

Atılım Güneş Baydin
University of Oxford

Carina Prunkl
University of Oxford

Olexandr Isayev
Carnegie Mellon University

Erik Peterson
Carnegie Mellon University

Peter L. McMahon
Cornell University

Jakob H. Macke
University of Tübingen

Kyle Cranmer
New York University

Jiaxin Zhang
Oak Ridge National Lab

Haruko Wainwright
Lawrence Berkeley National Lab

Adi Hanuka
SLAC National Accelerator Lab

Samuel Assefa
US Bank AI Innovation

Stephan Zheng
Salesforce Research

Manuela Veloso
JPM AI Research

Avi Pfeffer
Charles River Analytics

ABSTRACT

The original "Seven Motifs" set forth a roadmap of essential methods for the field of scientific computing, where a motif is an algorithmic method that captures a pattern of computation and data movement.[1] We present the *Nine Motifs of Simulation Intelligence*, a roadmap for the development and integration of the essential algorithms necessary for a merger of scientific computing, scientific simulation, and artificial intelligence. We call this merger simulation intelligence (SI), for short. We argue the motifs of simulation intelligence are interconnected and interdependent, much like the components within the layers of an operating system. Using this metaphor, we explore the nature of each layer of the simulation intelligence "operating system" stack (SI-stack) and the motifs therein:

1. Multi-physics and multi-scale modeling
2. Surrogate modeling and emulation
3. Simulation-based inference
4. Causal modeling and inference
5. Agent-based modeling
6. Probabilistic programming
7. Differentiable programming
8. Open-ended optimization
9. Machine programming

We believe coordinated efforts between motifs offers immense opportunity to accelerate scientific discovery, from solving inverse problems in synthetic biology and climate science, to directing nuclear energy experiments and predicting emergent behavior in socioeconomic settings. We elaborate on each layer of the SI-stack, detailing the state-of-art methods, presenting examples to highlight challenges and opportunities, and advocating for specific ways to advance the motifs and the synergies from their combinations. Advancing and integrating these technologies can enable a robust and efficient hypothesis–simulation–analysis type of scientific method, which we introduce with several use-cases for human-machine teaming and automated science.

Keywords: Simulation; Artificial Intelligence; Machine Learning; Scientific Computing; Physics-infused ML; Inverse Design; Human-Machine Teaming; Optimization; Causality; Complexity; Open-endedness

[*]lavin@simulation.science (ISI & Pasteur Labs)

[1]We eschew the original term "dwarf" for the more appropriate "motif" in this paper, and encourage the field to follow suit.

Preprint. Under review.

Contents

Introduction **3**

Simulation Intelligence Motifs **4**

The Modules **5**

1. MULTI-PHYSICS & MULTI-SCALE MODELING 5
2. SURROGATE MODELING & EMULATION 11
3. SIMULATION-BASED INFERENCE 17
4. CAUSAL REASONING 22
5. AGENT-BASED MODELING 28

The Engine **33**

6. PROBABILISTIC PROGRAMMING 33
7. DIFFERENTIABLE PROGRAMMING 40

The Frontier **45**

8. OPEN-ENDED OPTIMIZATION 45
9. MACHINE PROGRAMMING 48

Simulation Intelligence Themes **51**

INVERSE-PROBLEM SOLVING 51
UNCERTAINTY REASONING 54
INTEGRATIONS 55
HUMAN-MACHINE TEAMING 63

Simulation Intelligence in Practice **64**

Data-intensive science and computing 65
Accelerated computing 68
Domains for use-inspired research 70

Discussion **70**

Honorable mention motifs 70
Conclusion 74

Introduction

Simulation has become an indispensable tool for researchers across the sciences to explore the behavior of complex, dynamic systems under varying conditions [1], including hypothetical or extreme conditions, and increasingly tipping points in environments such as climate [2, 3, 4], biology [5, 6], sociopolitics [7, 8], and others with significant consequences. Yet there are challenges that limit the utility of simulators (and modeling tools broadly) in many settings. First, despite advances in hardware to enable simulations to model increasingly complex systems, computational costs severely limit the level of geometric details, complexity of physics, and the number of simulator runs. This can lead to simplifying assumptions, which often render the results unusable for hypothesis testing and practical decision-making. In addition, simulators are inherently biased as they simulate only what they are programmed to simulate; sensitivity and uncertainty analyses are often impractical for expensive simulators; simulation code is composed of low-level mechanistic components that are typically non-differentiable and lead to intractable likelihoods; and simulators can rarely integrate with real-world data streams, let alone run online with live data updates.

Recent progress with artificial intelligence (AI) and machine learning (ML) in the sciences has advanced methods towards several key objectives for AI/ML to be useful in sciences (beyond discovering patterns in high-dimensional data). These advances allow us to import priors or domain knowledge into ML models and export knowledge from learned models back to the scientific domain; leverage ML for numerically intractable simulation and optimization problems, as well as maximize the utility of real-world data; generate myriads of synthetic data; quantify and reason about uncertainties in models and data; and infer causal relationships in the data.

It is at the intersection of AI and simulation sciences where we can expect significant strides in scientific experimentation and discovery, in essentially all domains. For instance, the use of neural networks to accelerate simulation software for climate science [9], or multi-agent reinforcement learning and game theory towards economic policy simulations [10]. Yet this area is relatively nascent and disparate, and a unifying holistic perspective is needed to advance the intersection of AI and simulation sciences.

This paper explores this perspective. We lay out the methodologies required to make significant strides in simulation and AI for science, and how they must be fruitfully combined. The field of scientific computing was at a similar inflection point when Phillip Colella in 2004 presented to DARPA the *"Seven Dwarfs" for Scientific Computing*, where each of the seven represents an algorithmic method that captures a pattern of computation and data movement [11, 12, 13].[ii] For the remainder of this paper, we choose to replace a potentially insensitive term with "motif", a change we suggest for the field going forward.

The motifs nomenclature has proved useful for reasoning at a high level of abstraction about the behavior and requirements of these methods across a broad range of applications, while decoupling these from specific implementations. Even more, it is an understandable vocabulary for talking across disciplinary boundaries. Motifs also provide "anti-benchmarks": not tied to narrow performance or code artifacts, thus encouraging innovation in algorithms, programming languages, data structures, and hardware [12]. Therefore the motifs of scientific computing provided an explicit roadmap for R&D efforts in numerical methods (and eventually parallel computing) in sciences.

In this paper, we similarly define the *Nine Motifs of Simulation Intelligence*, classes of complementary algorithmic methods that represent the foundation for synergistic simulation and AI technologies to advance sciences; *simulation intelligence (SI)* describes a field that merges of scientific computing, scientific simulation, and artificial intelligence towards studying processes and systems *in silico* to better understand and discover *in situ* phenomena. Each of the SI motifs has momentum from the scientific computing and AI communities, yet must be pursued in concert and integrated in order to overcome the shortcomings of scientific simulators and enable new scientific workflows.

Unlike the older seven motifs of scientific computing, our SI motifs are not necessarily independent. Many of these are interconnected and interdependent, much like the components within the layers of an operating system. The individual modules can be combined and interact in multiple ways, gaining from this combination. Using this metaphor, we explore the nature of each layer of the "SI stack", the motifs within each layer, and the combinatorial possibilities available when they are brought together – the layers are illustrated in Fig. 1.

We begin by describing the core layers of the SI stack, detailing each motif within: the concepts, challenges, state-of-the-art methods, future directions, ethical considerations, and many motivating examples. As we traverse the SI stack, encountering the numerous modules and scientific workflows, we will ultimately be able to lay out how these advances will benefit the many users of simulation and scientific endeavors. Our discussion continues to cover important SI themes such as inverse problem solving and human-machine teaming, and essential infrastructure areas such as data engineering and accelerated computing.

[ii] The 2004 "Seven Motifs for Scientific Computing": Dense Linear Algebra, Sparse Linear Algebra, Computations on Structured Grids, Computations on Unstructured Grids, Spectral Methods, Particle Methods, and Monte Carlo [11].

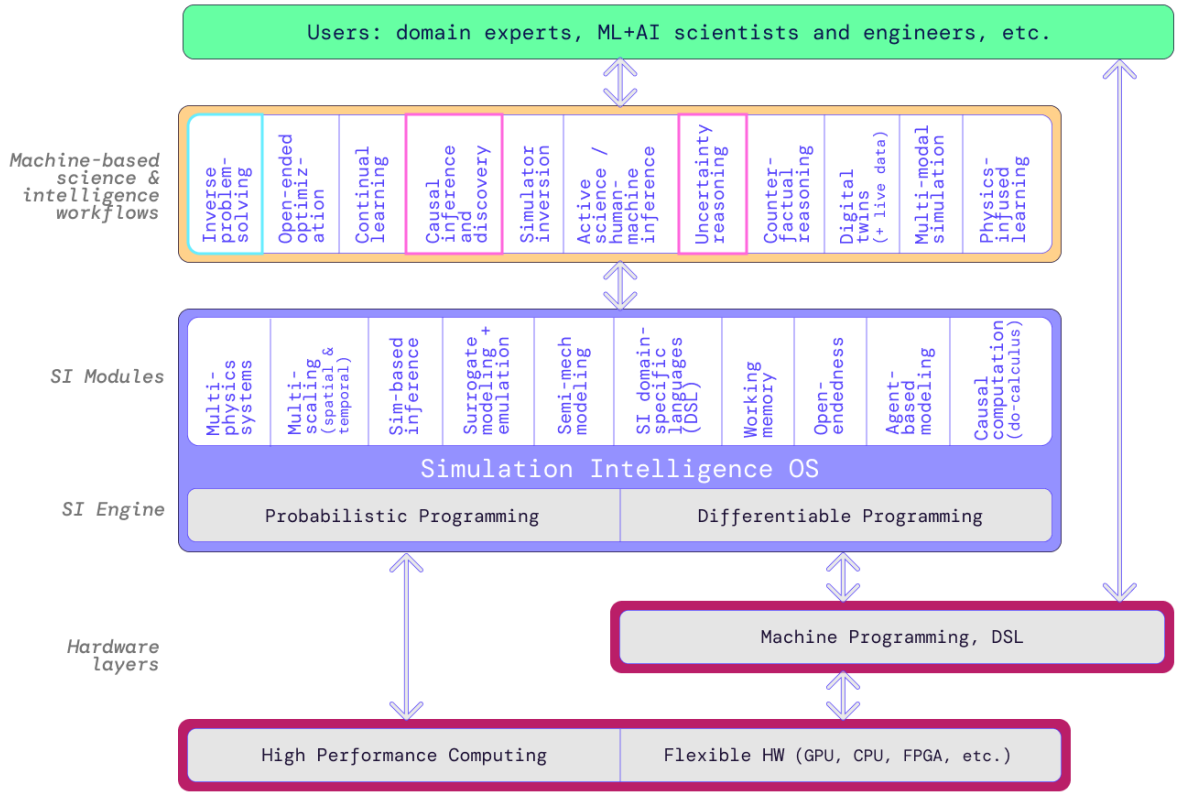

Figure 1: An operating system (OS) diagram, elucidating the relationships of the nine Simulation Intelligence motifs, their close relatives, such as domain-specific languages (DSL) [14] and working memory [15], and subsequent SI-based science workflows. The red, purple, orange, and green plates represent the hardware, OS, application, and user layers, respectively (from bottom to top). In the applications layer (orange plate) are some main SI workflows – notice some entries have a cyan outline, signifying machine-science classes that contain multiple workflows in themselves (see text for details), and entries with a pink outline denote statistical and ML methods that have been used for decades, but here in unique SI ways. In the text we generically refer to this composition as the "SI stack", although the mapping to an OS is a more precise representation of the SI layers and their interactions. We could have shown a simpler diagram with only the nine SI motifs, but the context of the broader stack that is enabled by their integration better elucidates the improved and new methods that can arise.

By pursuing research in each of these SI motifs, as well as ways of combining them together in generalizable software towards specific applications in science and intelligence, the recent trend of decelerating progress may be reversed [16, 17]. This paper aims to motivate the field and provide a roadmap for those who wish to work in AI and simulation in pursuit of new scientific methods and frontiers.

Simulation Intelligence Motifs

Although we define nine concrete motifs, they are not necessarily independent – many of the motifs are synergistic in function and utility, and some may be building blocks underlying others. We start by describing the "module" motifs, as in Fig. 1, followed by the underlying "engine" motifs: probabilistic and differentiable programming. We then describe the motifs that aim to push the frontier of intelligent machines: open-endedness and machine programming.

The Modules

Above the "engine" in the proverbial SI stack (Fig. 1) are "modules", each of which can be built in probabilistic and differentiable programming frameworks, and make use of the accelerated computing building blocks in the hardware layer. We start this section with several motifs that closely relate to the physics-informed learning topics most recently discussed above, and then proceed through the SI stack, describing how and why the module motifs compliment one another in myriad, synergistic ways.

1. MULTI-PHYSICS & MULTI-SCALE MODELING

Simulations are pervasive in every domain of science and engineering, yet are often done in isolation: climate simulations of coastal erosion do not model human-driven effects such as urbanization and mining, and even so are only consistent within a constrained region and timescale. Natural systems involve various types of physical phenomena operating at different spatial and temporal scales. For simulations to be accurate and useful they must support multiple physics and multiple scales (spatial and temporal). The same goes for AI & ML, which can be powerful for modeling multi-modality, multi-fidelity scientific data, but machine learning alone – based on data-driven relationships – ignores the fundamental laws of physics and can result in ill-posed problems or non-physical solutions. For ML (and AI-driven simulation) to be accurate and reliable in the sciences, methods must integrate multi-scale, multi-physics data and uncover mechanisms that explain the emergence of function.

Multi-physics Complex real-world problems require solutions that span a multitude of physical phenomena, which often can only be solved using simulation techniques that cross several engineering disciplines. Almost all practical problems in fluid dynamics involve the interaction between a gas and/or liquid with a solid object, and include a range of associated physics including heat transfer, particle transport, erosion, deposition, flow-induced-stress, combustion and chemical reaction. A multi-physics environments is defined by coupled processes or systems involving more than one simultaneously occurring physical fields or phenomena. Such an environment is typically described by multiple partial differential equations (PDEs), and tightly coupled such that solving them presents significant challenges with nonlinearities and time-stepping. In general, the more interacting physics in a simulation, the more costly the computation.

Multi-scale Ubiquitous in science and engineering, *cascades-of-scales* involve more than two scales with long-range spatio-temporal interactions (that often lack self-similarity and proper closure relations). In the context of biological and behavioral sciences, for instance, multi-scale modeling applications range from the molecular, cellular, tissue, and organ levels all the way to the population level; multi-scale modeling can enable researchers to probe biologically relevant phenomena at smaller scales and seamlessly embed the relevant mechanisms at larger scales to predict the emergent dynamics of the overall system [20]. Domains such as energy and synthetic biology require engineering materials at the nanoscale, optimizing multi-scale processes and systems at the macroscale, and even the discovery of new governing physico-chemical laws *across* scales. These scientific drivers call for a deeper, broader, and more integrated understanding of common multi-scale phenomena and scaling cascades. In practice, multi-scale modeling is burdened by computational inefficiency, especially with increasing complexity and scales; it is not uncommon to encounter "hidden" or unknown physics of interfaces, inhomogeneities, symmetry-breaking and other singularities.

Methods for utilizing information across scales (and physics that vary across space and time) are often needed in real-world settings, and with data from various sources: *Multi-fidelity modeling* aims to synergistically combine abundant, inexpensive, low-fidelity data and sparse, expensive, high-fidelity data from experiments and simulations. Multi-fidelity modeling is often useful in building efficient and robust surrogate models (which we detail in the surrogate motif section later) [21] – some examples include simulating the mixed convection flow past a cylinder [22] and cardiac electrophysiology [23].

In computational fluid dynamics (CFD), classical methods for multi-physics and multi-scale simulation (such as finite elements and pseudo-spectral methods) are only accurate if flow feature are all smooth, and thus meshes must resolve the smallest features. Consequently, direct numerical simulation for real-world systems such as climate and jet physics are impossible. It is common to use smoothed versions of the Navier Stokes equations to allow coarser meshes while sacrificing accuracy. Although successful in design of engines and turbo-machinery, there are severe limits to what can be accurately and reliably simulated – the resolution-efficiency tradeoff imposes a significant bottleneck. Methods for AI-driven acceleration and surrogate modeling could accelerate CFD and multi-physics multi-scale simulation by orders of magnitude. We discuss specific methods and examples below.

Physics-informed ML The newer class of *physics-informed machine learning* methods integrate mathematical physics models with data-driven learning. More specifically, making an ML method physics-informed amounts to introducing

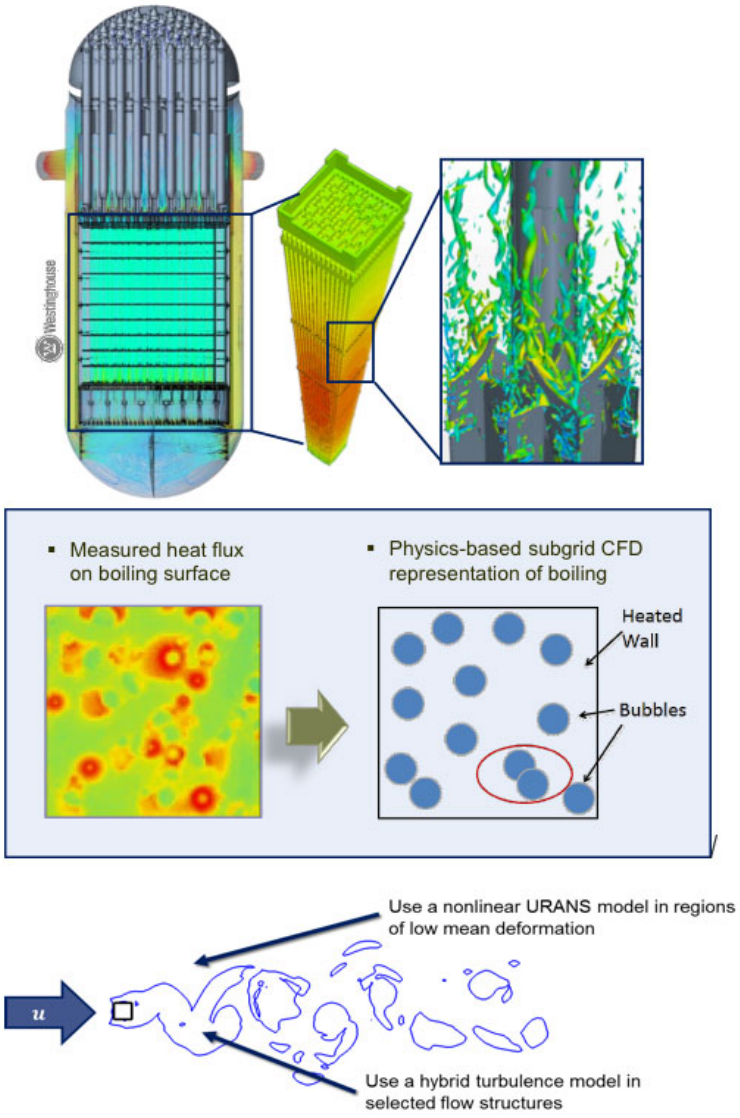

Figure 2: Diagram of the various physics and scales to simulate for turbulent flow within a nuclear reactor core – top-to-bottom zooming in reveals finer and finer scales, each with different physics to model. AI-driven computational fluid dynamics (CFD) can radically improve the resolution and efficiency of such simulations [18, 19].

appropriate observational, inductive, or learning biases that can steer or constrain the learning process to physically consistent solutions [24]:

1. *Observational biases* can be introduced via data that allows an ML system to learn functions, vector fields, and operators that reflect physical structure of the data.
2. *Inductive biases* are encoded as model-based structure that imposes prior assumptions or physical laws, making sure the physical constraints are strictly satisfied.
3. *Learning biases* force the training of an ML system to converge on solutions that adhere to the underlying physics, implemented by specific choices in loss functions, constraints, and inference algorithms.

A common problem template involves extrapolating from an initial condition obtained from noisy experimental data, where a governing equation is known for describing at least some of the physics. An ML method would aim to predict the latent solution $u(t, x)$ of a system at later times $t > 0$ and propagate the uncertainty due to noise in the initial data. A common use-case in scientific computing is reconstructing a flow field from scattered measurements (e.g., particle image velocimetry data), and using the governing Navier–Stokes equations to extrapolate this initial condition in time.

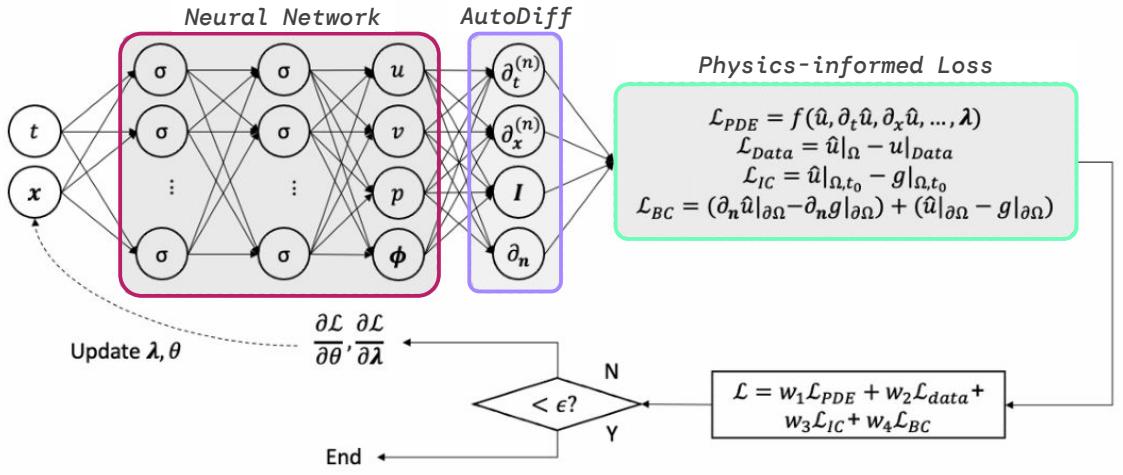

Figure 3: Diagram of a physics-informed neural network (PINN), where a fully-connected neural network (red), with time and space coordinates $(t, \mathbf{x})$ as inputs, is used to approximate the multi-physics solutions $\hat{u} = [u, v, p, \phi]$. The derivatives of $\hat{u}$ with respect to the inputs are calculated using automatic differentiation (purple; autodiff is discussed later in the SI engine section) and then used to formulate the residuals of the governing equations in the loss function (green), that is generally composed of multiple terms weighted by different coefficients. By minimizing the physics-informed loss function (i.e., the right-to-left process) we simultaneously learn the parameters of the neural network θ and the unknown PDE parameters λ. (Figure reproduced from Ref. [25])

For fluid flow problems and many others, Gaussian processes (GPs) present a useful approach for capturing the physics of dynamical systems. A GP is a Bayesian nonparametric machine learning technique that provides a flexible prior distribution over functions, enjoys analytical tractability, defines kernels for encoding domain structure, and has a fully probabilistic workflow for principled uncertainty reasoning [26, 27]. For these reasons GPs are used widely in scientific modeling, with several recent methods more directly encoding physics into GP models: *Numerical GPs* have covariance functions resulting from temporal discretization of time-dependent partial differential equations (PDEs) which describe the physics [28, 29], modified Matérn GPs can be defined to represent the solution to stochastic partial differential equations [30] and extend to Riemannian manifolds to fit more complex geometries [31], and the *physics-informed basis-function GP* derives a GP kernel directly from the physical model [32] – the latter method we elucidate in an experiment optimization example in the surrogate modeling motif.

With the recent wave of deep learning progress, research has developed methods for building the three physics-informed ML biases into nonlinear regression-based physics-informed networks. More specifically, there is increasing interest in *physics-informed neural networks (PINNs)*: deep neural nets for building surrogates of physics-based models described by partial differential equations (PDEs) [33, 34]. Despite recent successes (with some examples explored below) [35, 36, 37, 38, 39, 40], PINN approaches are currently limited to tasks that are characterized by relatively simple and well-defined physics, and often require domain-expert craftsmanship. Even so, the characteristics of many physical systems are often poorly understood or hard to implicitly encode in a neural network architecture.

Probabilistic graphical models (PGMs) [41], on the other hand, are useful for encoding *a priori* structure, such as the dependencies among model variables in order to maintain physically sound distributions. This is the intuition behind the promising direction of *graph-informed neural networks (GINN)* for multi-scale physics [42]. There are two main components of this approach (shown in Fig. 7): First, embedding a PGM into the physics-based representation to encode complex dependencies among model variables that arise from domain-specific information and to enable the generation of physically sound distributions. Second, from the embedded PGM identify computational bottlenecks intrinsic to the underlying physics-based model and replace them with an efficient NN surrogate. The hybrid model thus encodes a domain-aware physics-based model that synthesizes stochastic and multi-scale modeling, while computational bottlenecks are replaced by a fast surrogate NN whose supervised learning and prediction are further informed by the PGM (e.g., through structured priors). With significant computational advantages from surrogate modeling within GINN, we further explore this approach in the surrogate modeling motif later.

Modeling and simulation of complex nonlinear multi-scale and multi-physics systems requires the inclusion and characterization of uncertainties and errors that enter at various stages of the computational workflow. Typically we're concerned with two main classes of uncertainties in ML: *aleatoric* and *epistemic* uncertainties. The former quantifies system stochasticity such as observation and process noise, and the latter is model-based or subjective

uncertainty due to limited data. In environments of multiple scales and multiple physics, one should consider an additional type of uncertainty due the randomness of parameters of stochastic physical systems (often described by stochastic partial- or ordinary- differential equations (SPDEs, SODEs)). One can view this type of uncertainty arising from the computation of a sufficiently well-posed deterministic problem, in contrast to the notion of epistemic or aleatoric uncertainty quantification. The field of *probabilistic numerics* [43] makes a similar distinction, where the use of probabilistic modeling is to reason about uncertainties that arise strictly from the lack of information inherent in the solution of intractable problems such as quadrature methods and other integration procedures. In general the probabilistic numeric viewpoint provides a principled way to manage the parameters of numerical procedures. We discuss more on probabilistic numerics and uncertainty reasoning later in the SI themes section. The GINN can quantify uncertainties with a high degree of statistical confidence, while Bayesian analogs of PINNs are a work in progress [44]–one cannot simply plug in MC-dropout or other deep learning uncertainty estimation methods. The various uncertainties may be quantified and mitigated with methods that can utilize data-driven learning to inform the original systems of differential equations. We define this class of physics-*infused* machine learning later in this section and in the next motif.

The synergies of mechanistic physics models and data-driven learning are brought to bear when physics-informed ML is built with differentiable programming (one of the engine motifs), which we explore in the first example below.

Examples

Accelerated CFD via physics-informed surrogates and differentiable programming Kochkov et al. [18] look to bring the advantages of semi-mechanistic modeling and differentiable programming to the challenge of complex computational fluid dynamics (CFD). The Navier Stokes (NS) equations describe fluid dynamics well, yet in cases of multiple physics and complex dynamics, solving the equations at scale is severely limited by the computational cost of resolving the smallest spatiotemporal features. Approximation methods can alleviate this burden, but at the cost of accuracy. In Kochkov et al, the components of traditional fluids solvers most affected by the loss of resolution are replaced with better performing machine-learned alternatives (as presented in the semi-mechanistic modeling section later). This AI-driven solver algorithm is represented as a differentiable program with the neural networks and the numerical methods written in the JAX framework [45]. JAX is a leading framework for differentiable programming, with reverse-mode automatic differentiation that allows for end-to-end gradient based optimization of the entire programmed algorithm. In this CFD use-case, the result is an algorithm that maintains accuracy while using 10x coarser resolution in each dimension, yielding an 80-fold improvement in computation time with respect to an advanced numerical method of similar accuracy.

Related approaches implement PINNs without the use of differentiable programming, such as TF-Net for modeling turbulent flows with several specially designed U-Net deep learning architectures [46]. A promising direction in this and other spatiotemporal use-cases is *neural operator learning*: using NNs to learn mesh-independent, resolution-invariant solution operators for PDEs. To achieve this, Li et al. [47] use a Fourier layer that implements a Fourier transform, then a linear transform, and an inverse Fourier transform for a convolution-like operation in a NN. In a Bayesian inverse experiment, the Fourier neural operator acting as a surrogate can draw MCMC samples from the posterior of initial NS vorticity given sparse, noisy observations in 2.5 minutes, compared to 18 hours for the traditional solver.

Multi-physics and HPC simulation of blood flow in an intracranial aneurysm *SimNet* is an AI-driven multi-physics simulation framework based on neural network solvers–more specifically it approximates the solution to a PDE by a neural network [19]. SimNet improves on previous NN-solvers to take on the challenge of gradients and discontinuities introduced by complex geometries or physics. The main novelties are the use of Signed Distance Functions for loss weighting, and integral continuity planes for flow simulation. An intriguing real-world use case is simulating the flow inside a patient-specific geometry of an aneurysm, as shown in Fig. A. It is particularly challenging to get the flow field to develop correctly, especially inside the aneurysm sac. Building SimNet to support multi-GPU and multi-node scaling provides the computational efficiency necessary for such complex geometries. There's also optimization over repetitive trainings, such as training for surrogate-based design optimization or uncertainty quantification, where transfer learning reduces the time to convergence for neural network solvers. Once a model is trained for a single geometry, the trained model parameters are transferred to solve a different geometry, without having to train on the new geometry from scratch. As shown in Fig. B, transfer learning accelerates the patient-specific intracranial aneurysm simulations.

Raissi et al. [48] similarly approach the problem of 3D physiologic blood flow in a patient-specific intracranial aneurysm, but implementing a physics-informed NN technique: the Hidden Fluid Mechanics approach that uses autodiff (i.e. within differentiable programming) to simultaneously exploit information from the Navier Stokes equations of fluid dynamics and the information from flow visualization snapshots. Continuing this work has high potential for the robust

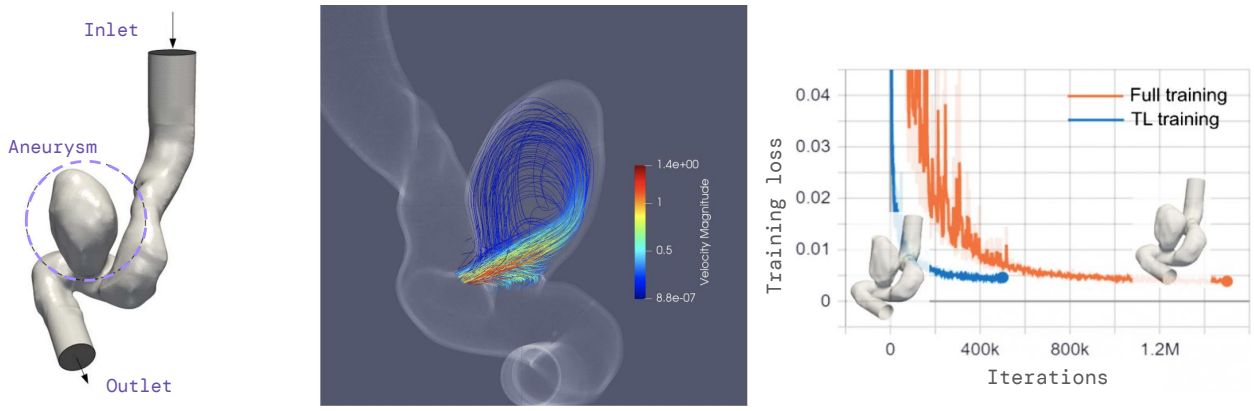

Figure 4: SimNet simulation results for the aneurysm problem [19]. Left: patient-specific geometry of an intracranial aneurysm. Center: Streamlines showing accurate flow field simulation inside the aneurysm sac. Right: Transfer learning within the NN-based simulator accelerates the computation of patient-specific geometries.

and data-efficient simulation in physical and biomedical applications. Raissi et al. effectively solve this as an inverse problem using blood flow data, while SimNet approaches this as a forward problem without data. Nonetheless SimNet has potential for solving inverse problems as well (within the inverse design workflow of Fig. 33 for example).

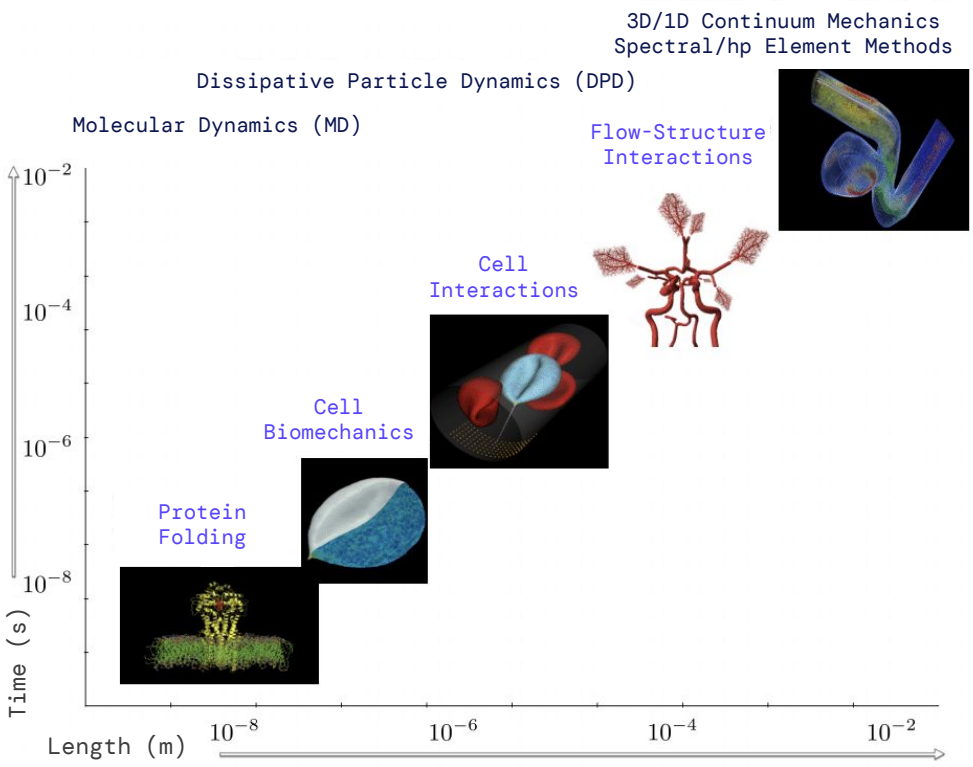

Figure 5: A cascade of spatial and temporal scales governing key biophysical mechanisms in brain blood flow requires effective multi-scale modeling and simulation techniques (inspired by [49]). Listed are the standard modeling approaches for each spatial and temporal regime, where an increase in computational demands is inevitable as one pursues an integrated resolution of interactions in finer and finer scales. SimNet [19], JAX MD [50], and related works with the motifs will help mitigate these computational demands and more seamlessly model dynamics across scales.

Learning quantum chemistry Using modern algorithms and supercomputers, systems containing thousands of interacting ions and electrons can now be described using approximations to the physical laws that govern the world on the atomic scale, namely the Schrödinger equation [51]. Chemical-simulation can allow for the properties of a compound to be anticipated (with reasonable accuracy) before synthesizing it in the laboratory. These computational chemistry applications range from catalyst development for greenhouse gas conversion, materials discovery for energy harvesting and storage, and computer-assisted drug design [52]. The potential energy surface is the central quantity of interest in the modeling of molecules and materials, computed by approximations to the time-independent Schrödinger equation. Yet there is a computational bottleneck: high-level wave function methods have high accuracy, but are often too slow for use in many areas of chemical research. The standard approach today is density functional theory (DFT) [53], which can yield results close to chemical accuracy, often on the scale of minutes to hours of computational time. DFT has enabled the development of extensive databases that cover the calculated properties of known and hypothetical systems, including organic and inorganic crystals, single molecules, and metal alloys [54, 55, 56]. In high-throughput applications, often used methods include force-field (FF) and semi-empirical quantum mechanics (SEQM), with runtimes on the order of fractions of a second, but at the cost of reliability of the predictions [57]. ML methods can potentially provide accelerated solvers without loss of accuracy, but to be reliable in the chemistry space the model must capture the underlying physics, and well-curated training data covering the relevant chemical problems must be available – Butler et al. [51] provide a list of publicly accessible structure and property databases for molecules and solids. Symmetries in geometries or physics, or *equivariance*, defined as the property of being independent of the choice of reference frame, can be exploited to this end. For instance, a message passing neural network [58] called OrbNet encodes a molecular system in graphs based on features from a quantum calculation that are low cost by implementing symmetry-adapted atomic orbitals [57, 59]. The authors demonstrate effectiveness of their equivariant approach on several organic and biological chemistry benchmarks, with accuracies on par to that of modern DFT functionals, providing a far more efficient drop-in replacement for DFT energy predictions. Additional approaches that constraining NNs to be symmetric under these geometric operations have been successfully applied in molecular design [60] and quantum chemistry [61].

Future directions

In general, machine learned models are ignorant of fundamental laws of physics, and can result in ill-posed problems or non-physical solutions. This effect is exacerbated when modeling the interplay of multiple physics or cascades of scales. Despite recent successes mentioned above, there is much work to be done for multi-scale and multi-physics problems. For instance, PINNs can struggle with high-frequency domains and frequency bias [62], which is particularly problematic in multi-scale problems [63]. Improved methods for learning multiple physics simultaneously are also needed, as the training can be prohibitively expensive – for example, a good approach with current tooling is training a model for each field separately and subsequently learning the coupled solutions through either a parallel or a serial architecture using supervised learning based on additional data for a specific multi-physics problem.

In order to approach these challenges in a collective and reproducible way, there is need to create *open benchmarks for physics-informed ML*, much like other areas of ML community such as computer vision and natural language processing. Yet producing quality benchmarks tailored for physics-informed ML can be more challenging:

1. To benchmark physics-informed ML methods, we additionally need the proper parameterized physical models to be explicitly included in the databases.
2. Many applications in physics and chemistry require full-field data, which cannot be obtained experimentally and/or call for significant compute.
3. Often different, problem-specific, physics-based evaluation methods are necessary, for example the metrics proposed in [64] for scoring physical consistency and [65] for scoring spatiotemporal predictions.

An overarching benchmarking challenge but also advantageous constraint is the multidisciplinary nature of physics-informed ML: there must be multiple benchmarks in multiple domains, rather than one benchmark to rule them all, which is a development bias that ImageNet has put on the computer vision field the past decade. And because we have domain knowledge and numerical methods for the underlying data generating mechanisms and processes in physical and life sciences, we have an opportunity to quantify robustly the characteristics of datasets to better ground the performances of various models and algorithms. This is contrast to the common practice of naïve data gathering to compile massive benchmark datasets for deep learning – for example, scraping the internet for videos to compose a benchmark dataset for human action recognition (deepmind.com/research/open-source/kinetics) – where not only are the underlying statistics and causal factors *a priori* unknown, the target variables and class labels are non-trivial to define and can lead to significant ethical issues such as dataset biases that lead to model biases, which in some cases can propagate harmful assumptions and stereotypes.

Further we propose to broaden the class of methods beyond *physics-informed*, to ***physics-infused machine learning***. The former is unidirectional (physics providing constraints or other information to direct ML methods), whereas the latter is bidirectional, including approaches that can better synergize the two computational fields. For instance, for systems with partial information, methods in physics-infused ML can potentially compliment known physical models to learn missing or misunderstood components of the systems. We specifically highlight one approach named *Universal Differential Equations* [66] in the surrogate modeling motif next.

Physics-infused ML can enable many new simulation tools and use-cases because of this ability to integrate physical models and data within a differentiable software paradigm. JAX and SimNet are nice examples, each enabling physics-informed ML methods for many science problems and workflows. Consider, for instance, the JAX fluid dynamics example above: another use-case in the same DP framework is JAX MD [50] for performing differentiable physics simulations with a focus on molecular dynamics.

Another exciting area of future multi-physics multi-scale development is inverse problem solving. We already mentioned the immense acceleration in the CFD example above with Fourier Neural Operators [47]. Beyond efficiency gains, physics-infused ML can take on applications with inverse and ill-posed problems which are either difficult or impossible to solve with conventional approaches, notably quantum chemistry: A recent approach called FermiNet [67] takes a dual physics-informed learning approach to solving the many-electron Schrodinger equation, where both inductive bias and learning bias are employed. The advantage of physics-infused ML here is eliminating extrapolation problems with the standard numerical approach, which is a common source of error in computational quantum chemistry. In other domains such as biology, biomedicine, and behavioral sciences, focus is shifting from solving forward problems based on sparse data towards solving inverse problems to explain large datasets [21]. The aim is to develop multi-scale simulations to infer the behavior of the system, provided access to massive amounts of observational data, while the governing equations and their parameters are not precisely known. We further detail the methods and importance of inverse problem solving with SI later in the Discussion section.

2. SURROGATE MODELING & EMULATION

A *surrogate model* is an approximation method that mimics the behavior of an expensive computation or process. For example, the design of an aircraft fuselage includes computationally intensive simulations with numerical optimizations that may take days to complete, making design space exploration, sensitivity analysis, and inverse modeling infeasible. In this case a computationally efficient surrogate model can be trained to represent the system, learning a mapping from simulator inputs to outputs. And Earth systems models (ESMs), for example, are extremely computationally expensive to run due to the large range of spatial and temporal scales and large number of processes being modeled. ESM surrogates can be trained on a few selected samples of the full, expensive simulations using supervised machine learning tools. In this simulation context, the aim of surrogate modeling (or *statistical emulation*) is to replace simulator code with a machine learning model (i.e., *emulator*) such that running the ML model to infer the simulator outputs is more efficient than running the full simulator itself. An emulator is thus a model of a model: a statistical model of the simulator, which is itself a mechanistic model of the world.

For surrogate modeling in the sciences, non-linear, nonparametric Gaussian processes (GP) [26] are typically used because of their flexibility, interpretability, and accurate uncertainty estimates [70]. Although traditionally limited to smaller datasets because of $O(N^3)$ computational cost of training (where N is the number of training data points), much work on reliable GP sparsification and approximation methods make them viable for real-world use [71, 72, 73, 74].

Neural networks (NNs) can also be well-suited to the surrogate modeling task as function approximation machines: A feedforward network defines a mapping $y = f(x; \theta)$ and learns the value of the parameters θ that result in the best function approximation. The *Universal Approximation Theorem* demonstrates that sufficiently large NNs can approximate any nonlinear function with a finite set of parameters [75, 76]. Although sufficient to represent any function, a NN layer may be unfeasibly large such that it may fail to learn and generalize correctly [77]. Recent work has shown that a NN with an infinitely wide hidden layer converges to a GP, representing the normal distribution over the space of functions.

In Fig. 6 we show an example of how a NN surrogate can be used as an emulator that encapsulates either the entire simulator or a specific part of the simulator. In the former, training is relatively straightforward because the loss function only has neural networks, and the trained network can be used towards inverse problem solving. However, we now have a black-box simulator: there is no interpretability of the trained network, and we cannot utilize the mechanistic components (i.e. differential equations of the simulator) for scientific analyses. In the case of the partial surrogate we have several advantages: the surrogate's number of parameters is reduced and thus the network is more stable (similar

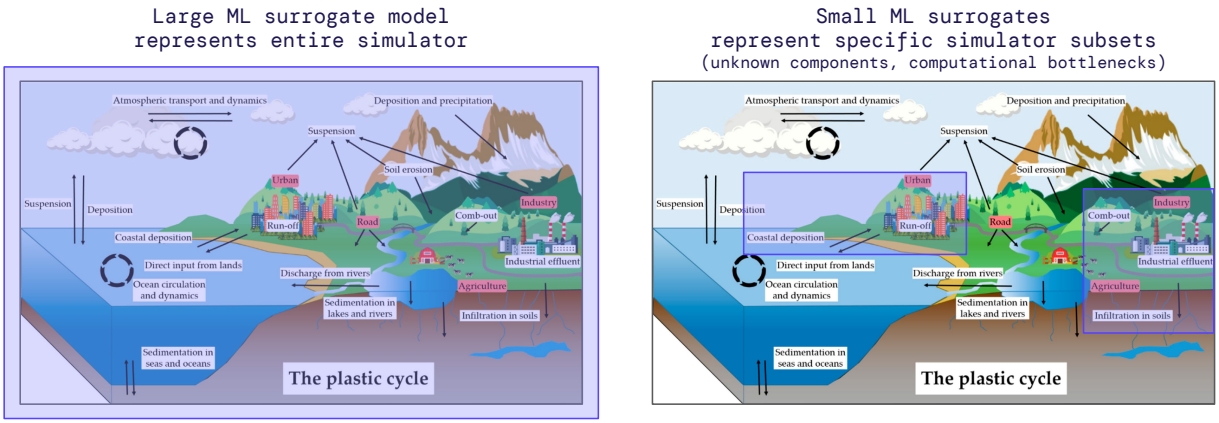

Figure 6: An example Earth system model (ESM) for the plastic cycle, with two variations of ML surrogates (purple screens): On the left, an ML surrogate learns the whole model. A variety of physics-infused ML methods can be applied – training is relatively straightforward because the loss function only has neural networks, and the trained network can be used towards inverse problem solving. However, we now have a black-box simulator: there is no interpretability of the trained network, and we cannot utilize the mechanistic components (i.e. differential equations of the simulator) for scientific analyses. On the right, two unknown portions of the model are learned by NN surrogates, while the remaining portions are represented by known mechanistic equations – possible with surrogate modeling approaches like UDE and GINN. This "partial surrogate" case has several advantages: the surrogate's number of parameters is reduced and thus the network is more stable (similar logic holds for nonparametric Gaussian process surrogate models), and the simulator retains all structural knowledge and the ability to run numerical analysis. The main challenge is that backpropagation of arbitrary scientific simulators is required, which we can address with the engine motif differentiable programming, producing learning gradients for arbitrary programs. The computational gain associated with the use of a hybrid surrogate-simulator cascades into a series of additional advantages including the possibility of simulating more scenarios towards counterfactual reasoning and epistemic uncertainty estimates, decreasing grid sizes, or exploring finer-scale parameterizations [68, 69].

logic holds for nonparametric GP surrogate models), and the simulator retains all structural knowledge and the ability to run numerical analysis. Yet the main challenge is that backpropagation of arbitrary scientific simulators is required – thus the significance of the differentiable programming motif we discussed earlier. The computational gain associated with the use of a hybrid surrogate-simulator cascades into a series of additional advantages including the possibility of simulating more scenarios towards counterfactual reasoning and epistemic uncertainty estimates, decreasing grid sizes, or exploring finer-scale parameterizations [68, 69].

Semi-mechanistic modeling It follows from the Universal Approximation Theorem that an NN can learn to approximate any sufficiently regular differential equation. The approach of recent neural-ODE methods [78] is to learn to approximate differential equations directly from data, but these can perform poorly when required to extrapolate [79]. More encouraging for emulation and scientific modeling, however, is to directly utilize mechanistic modeling simultaneously with NNs (or more generally, universal approximator models) in order to allow for arbitrary data-driven model extensions. The result is a *semi-mechanistic* approach, more specifically *Universal Differential Equations (UDE)* [66] where part of the differential equation contains a universal approximator model – we'll generally assume an NN is used for UDE in this paper, but other options include GP, Chebyshev expansion, or random forest.

UDE augments scientific models with machine-learnable structures for scientifically-based learning. This is very similar in motivation to the physics-informed neural nets (PINN) we previously discussed, but the implementation in general has a key distinction: a PINN is a deep learning model that become physics-informed because of some added physics bias (observational, inductive, or learning), whereas a UDE is a differential equation with one or more mechanistic components replaced with a data-driven model – this distinction is why we earlier defined physics-infused ML as the *bidirectional* influence of physics and ML.

Bayesian optimal experiment design A key aspect that makes emulators useful in scientific endeavors is that they allow us to reason probabilistically about *outer-loop decisions* such as optimization [80], data collection [81], and to use them to explain how uncertainty propagates in a system [82].

Numerous challenges in science and engineering can be framed as optimization tasks, including the maximization of reaction yields, the optimization of molecular and materials properties, and the fine-tuning of automated hardware protocols [83]. When we seek to optimize the parameters of a system with an expensive cost function f, we look to employ *Bayesian optimization (BO)* [80, 84, 85, 86] to efficiently explore the search space of solutions with a probabilistic surrogate model $\hat{f}$ rather than experimenting with the real system. Gaussian process models are the most common surrogates due to their flexible, nonparametric behavior. Various strategies to explore-exploit the search space can be implemented with acquisition functions to efficiently guide the BO search by estimating the utility of evaluating f at a given point (or parameterization). Often in science and engineering settings this provides the domain experts with a few highly promising candidate solutions to then try on the real system, rather than searching the intractably large space of possibilities themselves – for example, generating novel molecules with optimized chemical properties [87, 88], materials design with expensive physics-based simulations [89], and design of aerospace engineering systems [90].

Similarly, scientists can utilize BO for designing experiments such that the outcomes will be as informative as possible about the underlying process. *Bayesian optimal experiment design (BOED)* is a powerful mathematical framework for tackling this problem [91, 92, 93, 94], and can be implemented across disciplines, from bioinformatics [95] to pharmacology [96] to physics [97] to psychology [98]. In addition to *design*, there are also *control* methods in experiment optimization, which we detail in the context of a particle physics example below.

Examples

Simulation-based online optimization of physical experiments It is often necessary and challenging to design experiments such that outcomes will be as informative as possible about the underlying process, typically because experiments are costly or dangerous. Many applications such as nuclear fusion and particle acceleration call for online control and tuning of system parameters to deliver optimal performance levels – i.e., the *control* class of experiment design we introduced above.

In the case of particle accelerators, although physics models exist, there are often significant differences between the simulation and the real accelerator, so we must leverage real data for precise tuning. Yet we cannot rely on many runs with the real accelerator to tune the hundreds of machine parameters, and archived data does not suffice because there are often new machine configurations to try – a control or tuning algorithm must robustly find the optimum in a complex parameter space with high efficiency. With physics-infused ML, we can exploit well-verified mathematical models to learn approximate system dynamics from few data samples and thus optimize systems online and *in silico*. It follows that we can additionally look to optimize new systems without prior data.

For the online control of particle accelerators, Hanuka et al. [32] develop the *physics-informed basis-function GP*. To clarify what this model encompasses, we need to understand the several ways to build such a GP surrogate for BO of a physical system:

1. Data-informed GP using real experimental data
2. Physics-informed GP using simulated data
3. Basis-function GP from deriving a GP kernel directly from the physical model
4. Physics-informed basis-function GP as a combination of the above – methods 2 and 3 were combined in Hanuka et al.

The resulting physics-informed GP is more representative of the particle accelerator system, and performs faster in an online optimization task compared to routinely used optimizers (ML-based and otherwise). Additionally, the method presents a relatively simple way to construct the GP kernel, including correlations between devices – learning the kernel from simulated data instead of machine data is a form of kernel transfer learning, which can help with generalizability. Hanuka et al. interestingly point out that constructing the kernel from basis functions without using the likelihood function is a form of Gaussian process with *likelihood-free inference*, which is the regime of problems that simulation-based inference is designed for (i.e. the motif we discuss next).

This and similar physics-informed methods are emerging as a powerful strategy for *in silico* optimization of expensive scientific processes and machines, and further to enable scientific discovery by means of autonomous experimentation. For instance, the recent Gemini [99] and Golem [83] molecular experiment optimization algorithms, which are purpose-built for automated science workflows with SI: using surrogate modeling techniques for proxying expensive chemistry experiments, the BO and uncertainty estimation methods are designed for robustness to common scientific measurement challenges such as input variability and noise, proxy measurements, and systematic biases. Similarly, Shirobokov et al. [100] propose a method for gradient-based optimization of black-box simulators using local generative surrogates

that are trained in successive local neighborhoods of the parameter space during optimization, and demonstrate this technique in the optimization of the experimental design of the SHiP (Search for Hidden Particles) experiment proposed at CERN. These and other works of Alán Aspuru-Guzik et al. are good sources to follow in this area of automating science. There are potentially significant cause-effect implications to consider in these workflows, as we introduce in the causality motif later.

Multi-physics multi-scale surrogates for Earth systems emulation The climate change situation is worsening in accelerating fashion: the most recent decade (2010 to 2019) has been the costliest on record with the climate-driven economic damage reaching $2.98 trillion-US, nearly double the decade 2000–2009 [101]. The urgency for climate solutions motivates the need for modeling systems that are computationally efficient and reliable, lightweight for low-resource use-cases, informative towards policy- and decision-making, and cyber-physical with varieties of sensors and data modalities. Further, models need to be integrated with, and workflows extended to, climate-dependent domains such as energy generation and distribution, agriculture, water and disaster management, and socioeconomics. To this end, we and many others have been working broadly on *Digital Twin Earth (DTE)*, a catalogue of ML and simulation methods, datasets, pipelines, and tools for Earth systems researchers and decision-makers. In general, a *digital twin* is a computer representation of a real-world process or system – from large aircraft to individual organs. We define digital twin in the more precise sense of simulating the real physics and data-generating processes of an environment or system, with sufficient fidelity such that one can reliably run queries and experiments *in silico*.

Some of the main ML-related challenges for DTE include integrating simulations of multiple domains, geographies, and fidelities; not to mention the need to integrate real and synthetic data, as well as data from multiple modalities (such as fusing Earth observation imagery with on-the-ground sensor streams). SI methods play important roles in the DTE catalogue, notably the power of machine learned surrogates to accelerate existing climate simulators. Here we highlight one example for enabling lightweight, real-time simulation of coastal environments: Existing simulators for coastal storm surge and flooding are physics-based numerical models that can be extremely computationally expensive. Thus the simulators cannot be used for real-time predictions with high resolution, are unable to quantify uncertainties, and require significant computational infrastructure overhead only available to top national labs. To this end, Jiang et al. [102] developed physics-infused ML surrogates to emulate several of the main coastal simulators worldwide, NEMO [103] and CoSMoS [104]. Variations of the Fourier Neural Operator (FNO) [47] (introduced in the multi-physics motif) were implemented to produce upwards of 100x computational efficiency on comparable hardware.

This use-case exemplified a particularly thorny data preprocessing challenge that is commonplace working with spatiotemporal simulators and Digital Twin Earth: One of the coastal simulators to be emulated uses a standard grid-spaced representation for geospatial topology, which is readily computable with DFT in the FNO model, but another coastal simulator uses highly irregular grids that differ largely in scale, and further stacks these grids at varying resolutions. A preprocessing pipeline to regrid and interpolate the data maps was developed, along with substitute Fourier transform methods. It is our experience that many applications in DTE call for tailored solutions such as this – as a community we are lacking shared standards and formats for scientific data and code.

Hybrid PGM and NN for efficient domain-aware scientific modeling One can in general characterize probabilistic graphical models (PGM) [41] as structured models for encoding domain knowledge and constraints, contrasted with deep neural networks as data-driven function-approximators. The advantages of PGM have been utilized widely in scientfic ML [105, 106, 107, 108], and of course recent NN methods as discussed throughout this paper. *Graph-Informed Neural Networks (GINNs)* [42] are a new approach to incorporating the best of both worlds: PGMs incorporate expert knowledge, available data, constraints, etc. with physics-based models such as systems of ODEs and PDEs, while computationally intensive nodes in this hybrid model are replaced by learned features as NN surrogates. GINNs are particularly suited to enhance the computational workflow for complex systems featuring intrinsic computational bottlenecks and intricate physical relations variables. Hall et al. demonstrate GINN towards simulation-based decision-making in a multiscale model of electrical double-layer (EDL) supercapacitor dynamics. The ability for downstream decision-making is afforded by robust and reliable sensitivity analysis (due to the probabilistic ML approach), and orders of magnitude more computational efficiency means many hypotheses can be simulated and predicted posteriors quantified.

Auto-emulator design with neural architecture search Kasim et al. look to recent advances in *neural architecture search (NAS)* to automatically design and train a NN as an efficient, high-fidelity emulator, as doing this manually can be time-consuming and require significant ML expertise. NAS methods aim to learn a network topology that can achieve the best performance on a certain task by searching over the space of possible NN architectures given a set of NN primitives – see Elsken et al. [109] for a thorough overview.

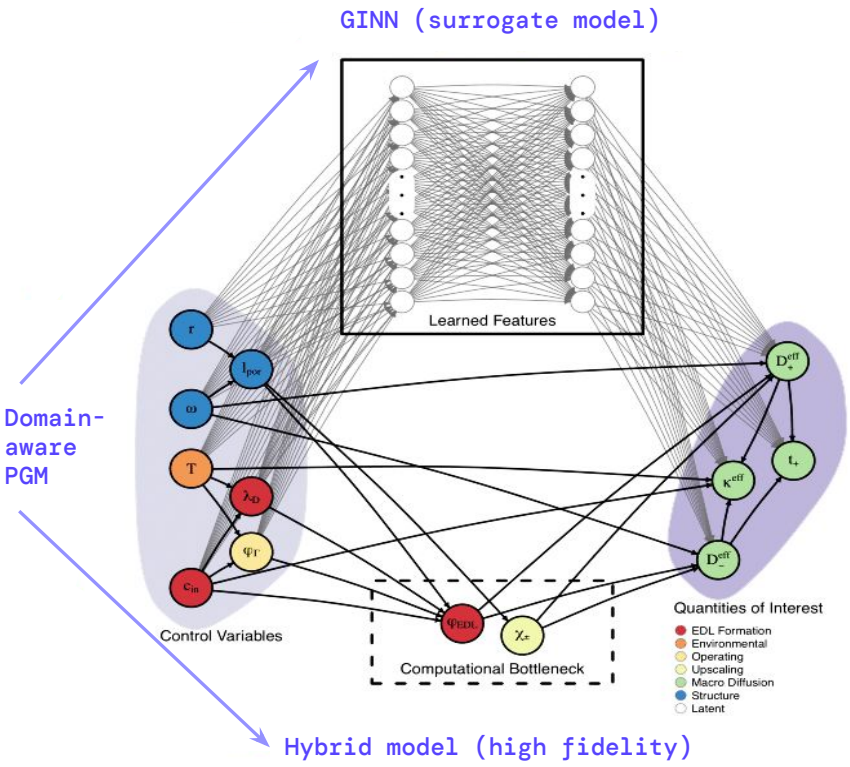

Figure 7: Graph-Informed Neural Networks (GINNs) [42] provide a computational advantage while maintaining the advantages of structured modeling with PGMs, by replacing computational bottlenecks with NN surrogates. Here a PGM encoding structured priors serves as input to both a Bayesian Network PDE (lower route) and a GINN (upper) for a homogenized model of ion diffusion in supercapacitors. A simple fully-connected NN is pictured, but in principle any architecture can work, for instance physics-informed methods that further enforce physical constraints.

The NAS results are promising for automated emulator construction: running on ten distinct scientific simulation cases, from fusion energy science [110, 111] to aerosol-climate [112] and oceanic [113] modeling, the results are reliably accurate output simulations with NN-based emulators that run thousands to billions times faster than the originals, while also outperforming other NAS based emulation approaches as well as manual emulator design. For example, a global climate model (GCM) simulation tested normally takes about 1150 CPU-hours to run [112], yet the emulator speedup is a factor of 110 million in direct comparison, and over 2 billion with a GPU — providing scientists with simulations on the order of seconds rather than days enables faster iteration of hypotheses and experiments, and potentially new experiments never before thought possible.

Also in this approach is a modified MC dropout method for estimating the predictive uncertainty of emulator outputs. Alternatively, we suggest pursuing Bayesian optimization-based NAS methods [114] for more principled uncertainty reasoning. For example, the former can flag when an emulator architecture is overconfident in its predictions, while the latter can do that *and* use the uncertainty values to dynamically adjust training parameters and search strategies.

The motivations of Kasim et al. for emulator-based accelerated simulation are same as we've declared above: enable rapid screening and ideas testing, and real-time prediction-based experimental control and optimization. The more we can optimize and automate the development *and* verification of emulators, the more efficiently scientists without ML expertise can iterate over simulation experiments, leading to more robust conclusions and more hypotheses to explore.

Deriving physical laws from data-driven surrogates Simulating complex dynamical systems often relies on governing equations conventionally obtained from rigorous first principles such as conservation laws or knowledge-based phenomenological derivations. Although non-trivial to derive, these symbolic or mechanistic equations are interpretable and understandable for scientists and engineers. NN-based simulations, including surrogates, are not interpretable in this way and can thus be challenging to use, especially in many cases where it is important the scientist or engineer understand the causal, data-generating mechanisms.

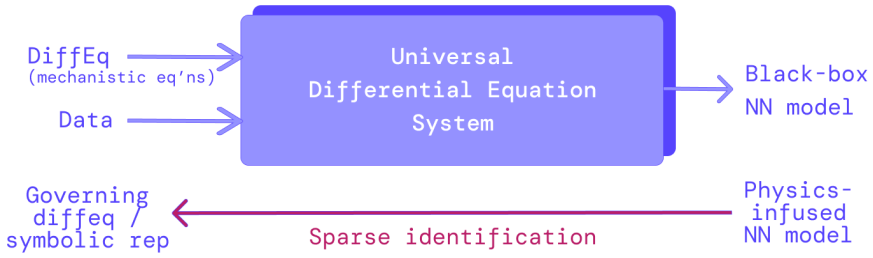

Figure 8: Illustrating the UDE forward process (top), where mechanistic equations are used with real data to produce a trained neural network (NN) model, followed by the inverse problem of recovering the governing equations in symbolic form (bottom).

Recent ML-driven advances have led approaches for *sparse identification of nonlinear dynamics (SINDy)* [115] to learn ODEs or PDEs from observational data. SINDy essentially selects dominant candidate functions from a high-dimensional nonlinear function space based on sparse regression to uncover ODEs that match the given data; one can think of SINDy as providing an ODE surrogate model. This exciting development has led to scientific applications from biological systems [116] to chemical processes [117] to active matter [118], as well as data-driven discovery of spatiotemporal systems governed by PDEs [119, 120]. Here we showcase two significant advances on SINDy utilizing methods described in the surrogate modeling motif and others:

1. **Synergistic learning deep NN surrogate and governing PDEs from sparse and independent data** – Chen et al. [121] present a novel physics-informed deep learning framework to discover governing PDEs of nonlinear spatiotemporal systems from scarce and noisy data accounting for different initial/boundary conditions (IBCs). Their approach integrates the strengths of deep NNs for learning rich features, automatic differentiation for accurate and efficient derivative calculation, and l_0 sparse regression to tackle the fundamental limitation of existing methods that scale poorly with data noise and scarcity. The special network architecture design is able to account for multiple independent datasets sampled under different IBCs, shown with simple experiments that should still be validated on more complex datasets. An alternating direction optimization strategy simultaneously trains the NN on the spatiotemporal data *and* determine the optimal sparse coefficients of selected candidate terms for reconstructing the PDE(s) – the NN provides accurate modeling of the solution and its derivatives as a basis for constructing the governing equation(s), while the sparsely represented PDE(s) in turn informs and constraints the DNN which makes it generalizable and further enhances the discovery. The overall semi-mechanistic approach – bottom-up (data-driven) and top-down (physics-informed) processes – is promising for ML-driven for scientific discovery.
2. **Sparse identification of missing model terms via Universal Differential Equations** – We earlier described several scenarios where an ML surrogate is trained for only part of the full simulator system, perhaps for the computationally inefficient or the unknown parts. This is also a use-case of the UDE, replacing parts of a simulator described by mechanistic equations with a data-driven NN surrogate model. Now consider we're at the end of the process of building a UDE (we have learned and verified an approximation for part of the causal generative model (i.e. a simulator)). Do we lose interpretability and analysis capabilities? With a knowledge-enhanced approach of the SINDy method we can sparse-identify the learned semi-mechanistic UDE back to mechanistic terms that are understandable and usable by domain scientists. Rackauckas et al. [66] modify the SINDy algorithm to apply to only subsets of the UDE equation in order to perform equation discovery specifically on the trained neural network components. In a sense this narrows the search space of potential governing equations by utilizing the prior mechanistic knowledge that wasn't replaced in training the UDE. Along with the UDE approach in general, this sparse identification method needs further development and validation with more complex datasets.

Future directions

The UDE approach has significant implications for use with simulators and physical modeling, where the underlying mechanistic models are commonly differential equations. By directly utilizing mechanistic modeling simultaneously with universal approximator models, UDE is a powerful semi-mechanistic approach allowing for arbitrary data-driven model extensions. In the context of simulators, this means a synergistic model of domain expertise and real-world data that more faithfully represents the true data-generating process of the system.

What we've described is a transformative approach for ML-augmented scientific modeling. That is,

1. Practitioner identifies known parts of a model and builds a UDE – when using probabilistic programming (an SI engine motif), this step can be done in a high-level abstraction where the user does not need to write custom inference algorithms.
2. Train an NN (or other surrogate model such as Gaussian process) to capture the missing mechanisms – one may look to NAS and Bayesian optimization approaches to do this in an automated, uncertainty-aware way.
3. The missing terms can be sparse-identified into mechanistic terms – this is an active area of research and much verification of this concept is needed, as mentioned in the example above.
4. Verify the recovered mechanisms are scientifically sane – for future work, how can we better enable this with human-machine teaming?
5. Verify quantitatively: extrapolate, do asymptotic analysis, run posterior predictive checks, predict bifurcations.
6. Gather additional data to validate[iii] the new terms.

Providing the tools for this semi-mechanistic modeling workflow can be immense for enabling scientists to make the best use of domain knowledge *and* data, and is precisely what the SI stack can deliver – notably with the differentiable programming and probabilistic programming "engine" motifs. Even more, building in a unified framework that's purpose-built for SI provides extensibility, for instance to integrate recent graph neural network approaches from Cranmer et al. [123] that can recover the governing equations in symbolic forms from learned physics-informed models, or the "AI Feynmann" [124, 125] approaches based on traditional fitting techniques in coordinate with neural networks that leverage physics properties such as symmetries and separability in the unknown dynamics function(s) – key features such as NN-equivariance and normalizing flows (and how they may fit into the stack) are discussed later.

3. SIMULATION-BASED INFERENCE

Numerical simulators are used across many fields of science and engineering to build computational models of complex phenomena. These simulators are typically built by incorporating scientific knowledge about the mechanisms which are known (or assumed) to underlie the process under study. Such mechanistic models have often been extensively studied and validated in the respective scientific domains. In complexity, they can range from extremely simple models that have a conceptual or even pedagogical flavor (e.g. the Lotka-Volterra equations describing predator-prey interactions in ecological systems and also economic theories [126] (expressed in Fig. 21)) to extremely detailed and expensive simulations implemented in supercomputers, e.g. whole-brain simulations [127].

A common challenge – across scientific disciplines and complexity of models – is the question of how to link such simulation-based models with empirical data. Numerical simulators typically have some parameters whose exact values are non *a priori*, and have to be inferred by data. For reasons detailed below, classical statistical approaches can not readily be applied to models defined by numerical simulators. The field of *simulation-based inference (SBI)* [1] aims to address this challenge, by designing statistical inference procedures that can be applied to complex simulators. Building on foundational work from the statistics community (see [128] for an overview), SBI is starting to bring together work from multiple fields – including, e.g., population genetics, neuroscience, particle physics, cosmology, and astrophysics – which are facing the same challenges and using tools from machine learning to address them. SBI can provide a unifying language, and common tools [129, 130, 131] and benchmarks [132] are being developed and generalized across different fields and applications.

Why is it so challenging to constrain numerical simulations by data? Many numerical simulators ave stochastic components, which are included either to provide a verisimilar model of the system under study if it is believed to be stochastic itself, or often also pragmatically to reflect incomplete knowledge about some components of the system. Linking such stochastic models with data falls within the domain of statistics, which aims to provide methods for constraining the parameters of a model by data, approaches for selecting between different model-candidates, and criteria for determining whether a hypothesis can be rejected on grounds of empirical evidence. In particular, statistical inference aims to determine which parameters – and combinations of parameters – are compatible with empirical data and (possibly) *a priori* assumptions. A key ingredient of most statistical procedures is the likelihood $p(x|\theta)$ of data x given parameters θ. For example, Bayesian inference characterizes parameters which are compatible both with data and prior by the posterior distribution $p(\theta|x)$, which is proportional to the product of likelihood and prior,

[iii] Note we use "verify" and "validate" specifically, as there's important difference between *verification and validation (V&V)*: verification asks "are we building the solution right?" whereas validation asks "are we building the right solution?" [122].

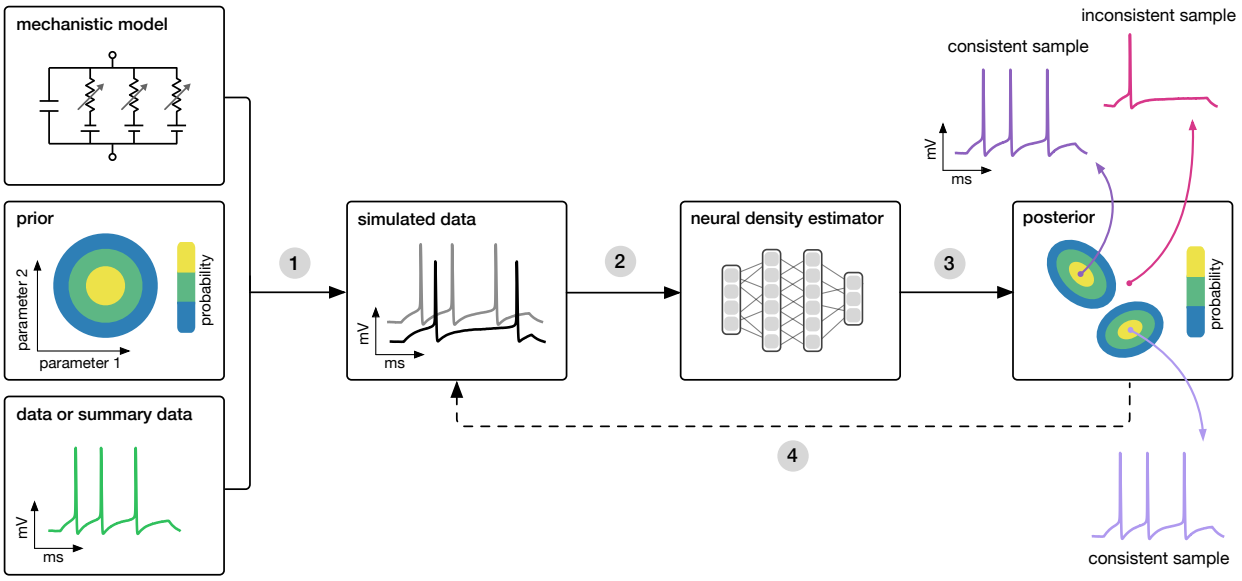

Figure 9: The goal of simulation-based inference (SBI) is to algorithmically identify parameters of simulation-based models which are compatible with observed data and prior assumptions. SBI algorithms generally take three inputs (left): A candidate mechanistic model (e.g. a biophysical neuron model), prior knowledge or constraints on model parameters, and observational data (or summary statistics thereof). The general process shown is to (1) sample parameters from the prior followed by simulating synthetic data from these parameters; (2) learn the (probabilistic) association between data (or data features) and underlying parameters (i.e., to learn statistical inference from simulated data) for which different SBI methods (discussed in the text) such as neural density estimation [133] can be used; (3) apply the learned model to empirical data to derive the full space of parameters consistent with the data and the prior, i.e. the *posterior distribution*. Posterior distributions may have complex shapes (such as multiple modes), and different parameter configurations may lead to data-consistent simulations. If needed, (4) an initial estimate of the posterior can be used to adaptively generate additional informative simulations. (Illustration from from [133])

$p(\theta|x) \propto p(x|\theta)p(\theta)$. Frequentist inference procedures typically construct confidence regions based on hypothesis tests, often using the likelihood ratio as test statistic.

However, for many simulation-based models, one can easily sample from the model (i.e., generate synthetic data $x \sim p(x|\theta)$) but *evaluating* the associated likelihoods can be computationally prohibitive – because, for instance, the same output x could result from a very large number of internal paths through the simulator, and integrating over all of them is prohibitive. More pragmatically, it might also be the case that the simulator is implemented in a "black-box" manner which does not provide access to its internal workings or states. If likelihoods can not be evaluated, most conventional inference approaches can not be used. The goal of simulation-based inference is to make statistical inference possible for so-called *implicit* models which allow generating simulated data, but not evaluation of likelihoods.

SBI is not a new idea. Simulation-based inference approaches have been studied extensively in statistics, typically under the heading of *likelihood-free inference*. An influential approach has been that of *Approximate Bayesian Computation* [128, 134, 135]. In its simplest form it consists of drawing parameter values from a proposal distribution, running the simulator for these parameters to generate synthetic outputs x $p(x|\theta)$, comparing these outputs against the observed data, and accepting the parameter values only if they are close to the observed data under some distance metric, $\|x - x_{\text{observed}}\| < \epsilon$. After following this procedure repeatedly, the accepted samples approximately follow the posterior. A second class of methods approximates the likelihood by sampling from the simulator and estimating the density in the sample space with kernel density estimation or histograms. This approximate density can then be used in lieu of the exact likelihood in frequentist or Bayesian inference techniques [136].

Both of these methods enable approximate inference in the likelihood-free setting, but they suffer from certain shortcomings: In the limit of a strict ABC acceptance criterion ($\epsilon \to 0$) or small kernel size, the inference results become exact, but the sample efficiency is reduced (the simulation has to be run many times). Relaxing the acceptance criterion or increasing the kernel size improves the sample efficiency, but reduces the quality of the inference results. The main challenge, however, is that these methods do not scale well to high-dimensional data, as the number of required simulations grows approximately exponentially with the dimension of the data x. In both approaches, the

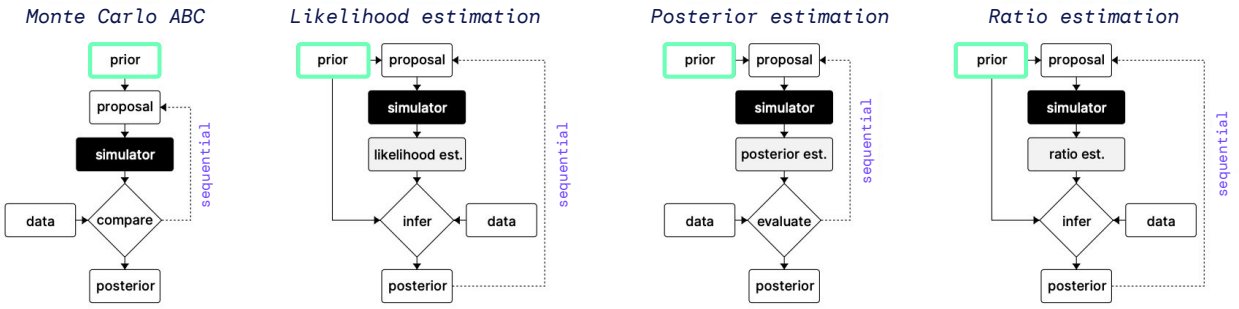

Figure 10: Various simulation-based inference workflows (or prototypes) are presented in Cranmer et al. [1]. Here we show four main workflows (or templates) of simulation-based inference: the left represents Approximate Bayesian Computation (ABC) approaches, and then to the right are three model-based approaches for approximating likelihoods, posteriors, and density ratios, respectively. Notice that all include algorithms that use the prior distribution to propose parameters (green), as well as algorithms for sequentially adapting the proposal (purple)—i.e., steps (1) and (4) shown in Fig. 9. (Figure reproduced from Ref. [132])

raw data is therefore usually first reduced to low-dimensional summary statistics. These are typically designed by domain experts with the goal of retaining as much information on the parameters θ as possible. In many cases, the summary statistics are not sufficient and this dimensionality reduction limits the quality of inference or model selection [137]. Recently, new methods for *learning* summary statistics in a fully [138, 139] or semi-automatic manner [140] are emerging, which might alleviate some of these limitations.

The advent of deep learning has powered a number of new simulation-based inference techniques. Many of these methods rely on the key principle of training a neural surrogate for the simulation. Such models are closely related to the emulator models discussed in the previous section, but not geared towards efficient sampling. Instead, we need to be able to access its likelihood [141, 142] (or the related likelihood ratio [143, 144, 145, 146, 147, 148]) or the posterior [149, 150, 151, 152]. After the surrogate has been trained, it can be used during frequentist or Bayesian inference instead of the simulation. On a high level, this approach is similar to the traditional method based on histograms or kernel density estimators [136], but modern ML models and algorithms allow it to scale to higher-dimensional and potentially structured data.

The impressive recent progress in SBI methods does not stem from deep learning alone. Another important theme is active learning: running the simulator and inference procedure iteratively and using past results to improve the proposal distribution of parameter values for the next runs [141, 142, 150, 151, 153, 154, 155, 156, 157, 158]. This can substantially improve the sample efficiency. Finally, in some cases simulators are not just black boxes, but we have access to (part of) their latent variables and mechanisms or probabilistic characteristics of their stack trace. In practice, such information can be made available through domain-specific knowledge or by implementing the simulation in a framework that supports differential or probabilistic programming– i.e., the SI engine. If it is accessible, such data can substantially improve the sample efficiency with which neural surrogate models can be trained, reducing the required compute [159, 160, 161, 162, 163]. On a high level, this represents a tighter integration of the inference engine with the simulation [164].

These components – neural surrogates for the simulator, active learning, the integration of simulation and inference – can be combined in different ways to define workflows for simulation-based inference, both in the Bayesian and frequentist setting. We show some example inference workflows in Fig. 10. The optimal choice of the workflow depends on the characteristics of the problem, in particular on the dimensionality and structure of the observed data and the parameters, whether a single data point or multiple i.i.d. draws are observed, the computational complexity of the simulator, and whether the simulator admits accessing its latent process.

Examples

Simulation-based inference techniques have been used to link models to data in a wide range of problems, reflecting the ubiquity of simulators across the sciences. In physics, SBI is useful from the smallest scales in particle colliders [165, 166, 167], where it allows us to measure the properties of the Higgs boson with a higher precision and less data, to the largest scales in the modeling of gravitational waves [168, 169], stellar streams [170], gravitational lensing [171], and the evolution of the universe [172]. These methods have also been applied to study the evolutionary dynamics of protein networks [173] and in yeast strains [174]. In neuroscience, they have, e.g., been used to estimate the properties of ion

channels from high-throughput voltage-clamp, and properties of neural circuits from observed rhythmic behaviour in the stomatogastric ganglion [133], to identify network models which can capture the dynamics of neuronal cultures [175], to study how the connectivity between different cell-types shapes dynamics in cortical circuits [176], and to identify biophysically realistic models of neurons in the retina [177, 178].

SBI is not restricted to the natural sciences. In robotics, for example, simulators are commonly used to generate training data on which the parameters of control- and policy-search algorithms are subsequently trained. A common challenge is the question of how policies trained on simulated data can subsequently be transferred to the real world, called "sim-to-real" transfer. SBI has been used to infer posterior-distributions over the parameters of simulators [179, 180] – the idea is that simulations from the posterior would yield more realistic data than 'hand-tuned' simulators, and that by simulating from the posterior, one would also cover a range of different domains (or distributions, environment variations, etc.) rather than just over-fit on a single parameter regime. For example, [181] used this approach to identify posteriors over hand-configuration, on which a multi-fingered robotic grasping algorithm was trained.

Here we will not be able to do the wide variety of example applications of SBI justice, and instead briefly zoom in on two particular use cases from particle physics and neuroscience.

Measuring the properties of the Higgs boson The experiments at the Large Hadron Collider probe the laws of physics at the smallest length scales accessible to humans. A key goal is to precisely measure how the Higgs boson interacts with other elementary particles: patterns in these interactions may provide evidence for physics beyond the "Standard Model", the established set of elementary particles and laws that govern them, and ultimately help us answer some of the open fundamental questions of elementary physics.

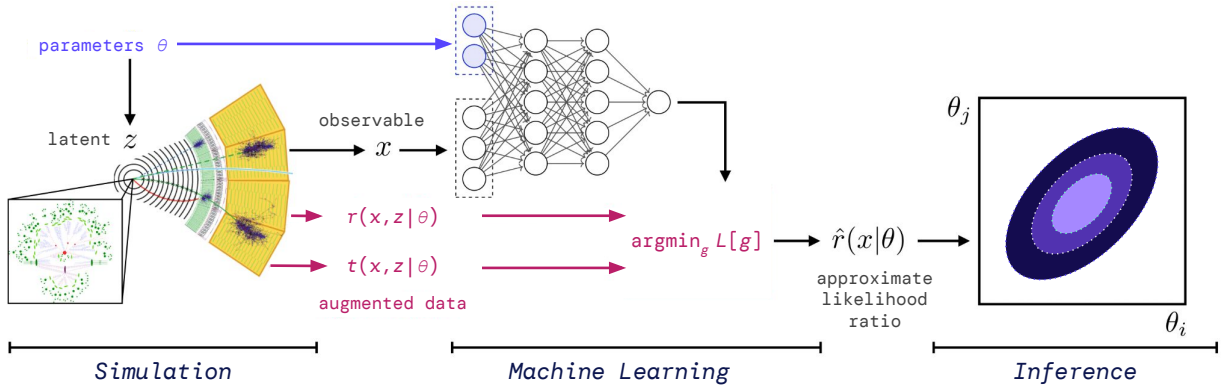

Figure 11: Schematic of our simulation-based inference workflow used in particle physics. (Reproduced from Ref. [165])

In this endeavour physicists rely on a chain of high-fidelity simulators for the interactions between elementary particles, their radiation and decay patterns, and the interactions with the detector. Owing to the large number of stochastic latent variables (state-of-the-art simulations involve multiple millions of random variables), the likelihood function of these simulators is intractable. Inference is traditionally made possible by first reducing the high-dimensional data to handcrafted summary statistics and estimating the likelihood with histograms, as discussed above, but popular summary statistics are often lossy, leading to less precise measurements.

Recently, particle physicists have begun developing and applying simulation-based inference methods driven by machine learning to this problem [165, 167, 182]. These methods on the one hand draw from the growing interdisciplinary toolbox of simulation-based inference and on the rapid progress in deep learning. On the other hand, they are tailored to the characteristics of this scientific measurement problem: they make use of the latent structure in particle physics simulators, are geared towards a large number of i. i. d. samples, and support inference in a frequentist framework, the established paradigm in particle physics. Multiple different algorithms have been proposed, but the basic strategy commonly consists of three steps (as we sketch in Fig. 11):

1. The chain of simulators is run for various values of the input parameters θ, for instance characterizing the interaction patterns between the Higgs boson and other elementary particles. These parameters are sampled from a proposal distribution, which needs to cover the region of interest in the parameter space (but does not have to be related to a Bayesian prior). For each simulated sample, the simulator inputs θ and outputs $x \sim p(x|\theta)$ are stored. The outputs x can consist of low-level data like energy deposits in the various detector components, but often it is more practical to parameterize them in form of preprocessed high-level variables

like the kinematic properties of reconstructed particles. In addition, certain additional latent variables can be stored that characterize the simulated process.

2. A machine learning model, often a multilayer perceptron [183], is trained on the simulated dataset, minimizing one of several proposed loss functions. They all have in common that (assuming sufficient capacity, infinite data, and perfect optimization) the network will converge to the likelihood ratio function $r(x|\theta) = p(x|\theta)/p_{\mathrm{ref}}(x)$, where $p_{\mathrm{ref}}(x)$ is a reference distribution (for instance the marginal likelihood or the likelihood for some value of the parameters). Some loss functions leverage the latent variables from the simulator process to improve the sample efficiency of the training.

3. The observed dataset consists of multiple i. i. d. events x_{obs}. The trained network is evaluated for each of these events. This provides a tractable and differentiable approximate likelihood function (up to an irrelevant overall constant), which is then used to define the maximum likelihood estimator $\hat{\theta}$ as well as confidence regions based on likelihood ratio tests, often relying on convenient asymptotic properties—in this frequentist approach, nuisance parameters through profiling [184].

First phenomenological studies on synthetic data have shown that these methods have the potential to substantially improve the precision of Higgs measurements at the LHC [165, 166, 167, 185, 186, 187, 188]. It remains a challenge to scale these analyses to real data and in particular to a realistic modeling of systematic uncertainties, which are usually modeled with thousands of nuisance parameters.

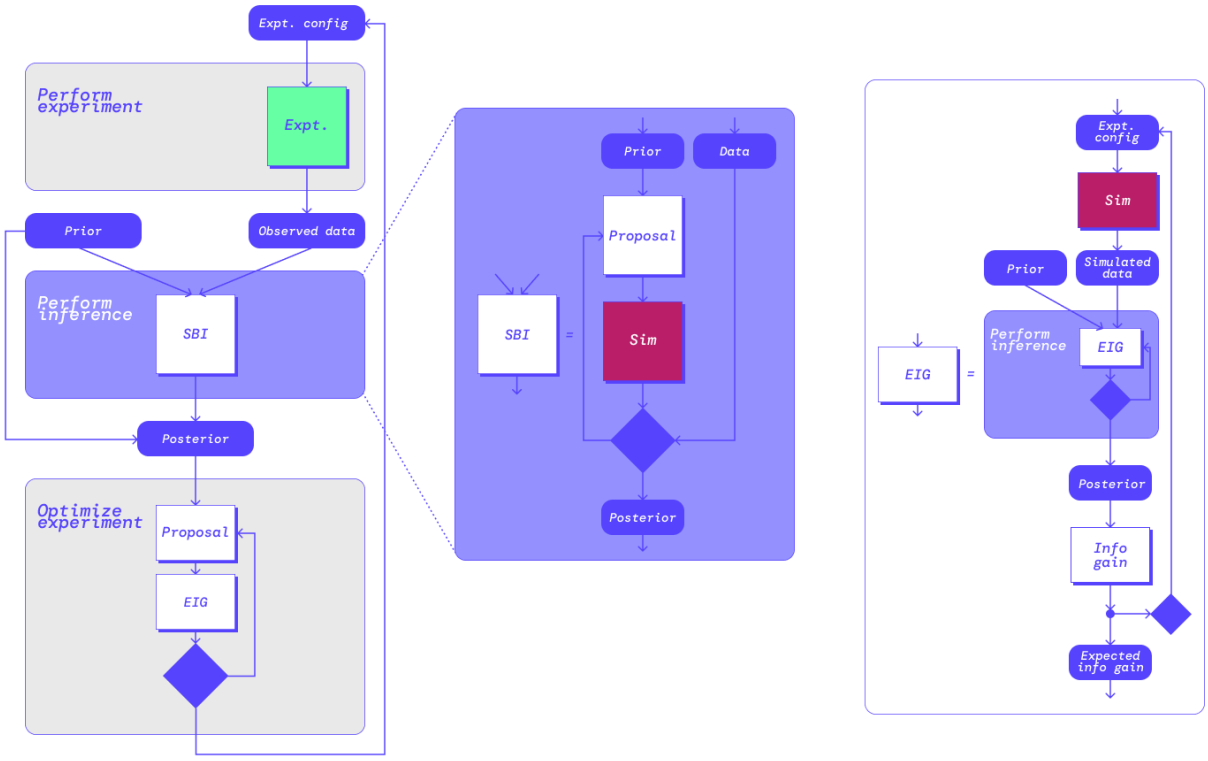

Figure 12: Workflows for "active science" with simulation-based inference methods [189], which we call *human-machine inference*. The scientist's inputs to the system are an external workflow that implements some experimental protocol (green Expt), and an external workflow that implements a simulator for those experiments, which depends on some theoretical parameters that we would like to infer (red Sim component). The center diagram double-clicks on the simulation-based inference (SBI) part, representing generic likelihood-free inference engines that enable statistical inference on the parameters of a theory that are implicitly defined by a simulator. The right diagram shows how the experiment is optimized with active learning and sequential design algorithms (namely Bayesian optimization, to balance exploration and exploitation to efficiently optimize an expensive black box objective function). Here the objective to optimize is expected information gain (EIG) over the hypothesis parameters.

Future directions

An intriguing direction to further develop SBI methods is towards *human-machine inference*: an efficient and automated loop of the scientific method (at least, for sufficiently well posed problems). Described in Fig. 12, the overall process is (1) perform experiment, (2) perform inference, (3) optimize experiment, and (4) repeat from (1). This approach provides an efficient means of physical experiments, particularly for expensive experiments in large search and hypothesis spaces, for instance the Large Hadron Collider at CERN [190]. This is one means by which simulation intelligence can broaden the scientific method. It will be fascinating to advance this SI-based science workflow by replacing the information gain objectives with open-ended optimization (discussed later in the SI engine motifs).

Simulation-based inference is a rapidly advancing class of methods, with progress fueled by newer programming paradigms such as probabilistic programming and differentiable programming. Cranmmer et al. [1] predict that "several domains of science should expect either a significant improvement in inference quality or the transition from heuristic approaches to those grounded in statistical terms tied to the underlying mechanistic model. It is not unreasonable to expect that this transition may have a profound impact on science."

4. CAUSAL REASONING

Standard approaches in statistical analysis and pattern recognition draw conclusions based on associations among variables and learned correlations in the data. Yet these approaches do not suffice when scientific queries are of *causal* nature rather than associative – there is marked distinction between correlations and *cause-effect* relationships. Such causal questions require some knowledge of the underlying data-generating process, and cannot be computed from the data alone, nor from the distributions that govern the data [191, 192]. Understanding causality is both the basis and the ultimate goal of scientific research in many domains – from genomics [193] and healthcare [194], to economics [195] and game theory [196], to Earth systems sciences [197] and more.

Abstractly we can consider a causal model to be qualitatively between a mechanistic model (described by differential equations, for example) and a statistical model (learned from data). That is, a mechanistic model is a description of the dynamics governing a system but is manually encoded by human experts, while a statistical model such as a neural network learns a mapping of variables from observational data but that only holds when the experimental conditions are fixed. Neither allows for causal queries. Causal learning lies between these two extremes, seeking to model the effects of interventions and distribution changes with a combination of data-driven learning and assumptions not already included in the statistical description of a system [198].

Reasoning about cause and effect with a causal model corresponds to formulating causal queries, which have been, organized by Pearl & Mackenzie [199] in a three-level hierarchy termed the *ladder of causation* [200] (Fig. 13):

1. *Observational* queries: seeing and observing. What can we tell about Y if we observe $X = x$?
2. *Interventional* queries: acting and intervening. What can we tell about Y if we do $X := x$?
3. *Counterfactual* queries: imagining, reasoning, and understanding. Given that $E = e$ actually happened, what would have happened to Y had we done $X := x$?

where X and Y are variables in the system of interest with events E. The difference between observation (conditional probability) and action (interventional calculus) is what motivated the development of causality [201]. More precisely the disagreement between interventional statements and conditional statements is defined as *confounding*: X and Y are confounded when the causal effect of action $X := x$ on Y does not coincide with the corresponding conditional probability.

It is common to approach the analysis of causal relationships with *counterfactual reasoning*: the difference between the outcomes if an action had been taken and if it had not been taken is defined as the causal effect of the action [192, 203]. Counterfactual problems involve reasoning about why things happened, imagining the consequences of different actions for which you never observe, and determining which actions would have achieved a desired outcome. A counterfactual question may be "would the virus have spread outside the city if the majority of adults were vaccinated one week sooner?" In practice this can be done with *structural causal models (SCMs)*, which give us a way to formalize the effect of hypothetical actions or interventions on the population within the assumptions of our model. An SCM, represented as a directed acyclic graph (DAG), is a collection of formal assumptions about how certain variables interact and compose the data generating process.

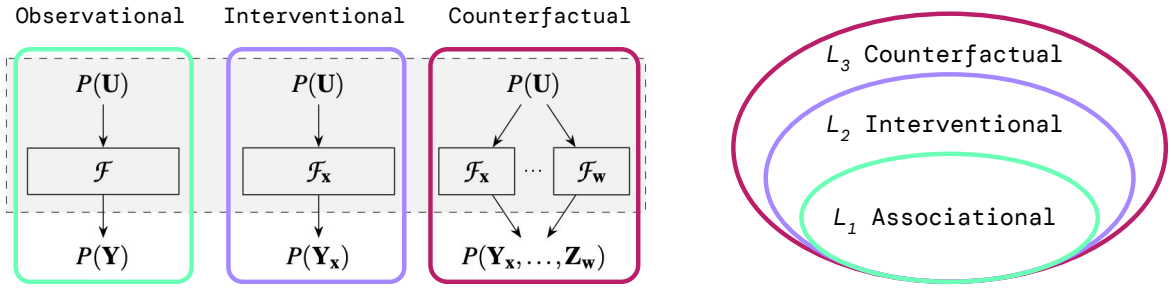

Figure 13: The three levels of the causal hierarchy: (1) associational, (2) interventional, and (3) counterfactual. Left: $P(U)$ describes the natural state of a system in domain U, and F represents the transformation to an observational world (green), an interventional world with modifying mechanism F_X (purple), and multiple counterfactual worlds with multiple modifying mechanisms $\{X, ..., W\}$ (red). Right: The three levels of the causal hierarchy separate in a measure-theoretic sense: data at one level almost always underdetermines data at higher levels. (Figure reproduced from Ref. [202])

Answers to counterfactual questions strongly depend on the specifics of the SCM; it is possible to construct two models that have identical graph structures, and behave identically under interventions, yet give different answers or potential outcomes [204]. Without an SCM, we can still obtain causal estimates based on counterfactuals by using the *potential outcomes framework*, or the Rubin-Neymann causal model [205, 206], which computes cause-effect estimates by considering the set of potential events or actions leading to an outcome Y.

A step below on the causal ladder are *interventional* questions, for instance "how does the probability of herd immunity change if the majority of adults are vaccinated?" Interventions in an SCM are made by modifying a subset of variable assignments in the model. Several types of interventions exist [207]: hard/perfect involves directly fixing a variables value, soft/imperfect corresponds to changing the conditional distribution, uncertain intervention implies the learner is unsure which mechanism/variable is affected by the intervention, and of course the no intervention base case, where only observational data is obtained from the SCM.

From the machine learning perspective, counterfactual reasoning is strictly harder than interventional as it requires imagining action rather than doing the action. And associative reasoning – that is, with a statistical model such as a neural network – is the simplest of all as it purely observes data (i.e., level 1 of the causal ladder). We illustrate these three levels of causality in Fig. 13. For full treatment please refer to Pearl's many works [191, 192], the causal inference survey by Yao et al. [208], Schölkopf et al. [198, 209] and the references therein for more details on causality in machine learning, and Guo et al. [210] for an overview of causal data science broadly. Here we focus on causality in scientific simulators.

Simulators provide virtual testbeds for the exploration of complex real-world scenarios, and, if built with the tooling for causal machine learning (specifically counterfactual reasoning), can provide "what-if" analyses in an efficient and cost-effective way. That is, each run of a simulator is a realization of a particular possible world, and counterfactual what-if claims can be studied by varying independent parameters of the model and re-running the simulation.

Simulation trace-based causality Given simulator execution traces in general consist of states, events, and actions, we can define causal relationships based on these entities – without this grounding, one cannot make reliable causal claims, which we discuss in the final example of this section. Herd & Miles [211] provide a practical formulation: event C causes event E in an actual trace π if and only if there exists a counterfactual trace π' that is similar enough to π which requires that neither C nor E happens in the same state in which they happened in π. This suggests counterfactual analysis can be performed by searching all traces produced by the simulator for counterfactual π', but this is clearly intractable for any non-trivial simulator. However, if we *intervene* on the trace during the simulation process, a variety of counterfactual traces can be generated with slight differences in events such that we may reason about the causal relationships between events. It follows that this approach is called *intervention-based counterfactual analysis*.

Simulation traces can be complex, amounting to several forms of causation based on the occurrence and omission of events in trace π [211]:

1. *Basic causation* – In order for event A to cause event B ($A \to B$), A's occurrence has to be both sufficient and necessary for B in π. That is, for the mathematically inclined reader, $A \to B \equiv (A \Rightarrow B) \wedge (B \Rightarrow A) \equiv A \Leftrightarrow B$.

2. *Prevention* – In order for event A to prevent event B, A's occurrence has to be both necessary and sufficient for B's omission. That is, $A \rightarrow B \equiv (A \Rightarrow \neg B) \wedge (\neg B \Rightarrow A) \equiv A \Leftrightarrow \neg B$.
3. *Transitivity* – If A causes B and B causes C then A causes C.
4. *Causation by omission*– If A's omission is both necessary and sufficient for B's occurrence and we accept that omissions are allowed to be causes, then A's omission can be considered to cause B's occurrence.
5. *Prevention by caused prevention*– It follows from the prior causal claims that if A causes B and B prevents C, then A prevents C.

The execution of a simulator results in one or more simulation traces, which represent various paths of traversing the state space given the underlying data-generating model, which is not unlike the execution traces resulting from probabilistic programming inference. By formalizing the sequence of simulator states as trace-based definition of causal dependencies, we have the computational and mathematical tooling to make causal inquiries, which we introduce in examples below. Further, with a simulator that produces a full posterior over the space of traces, as in probabilistic programming, we can attempt to draw causal conclusions with causal inference and discovery algorithms. This we introduce later in the motifs integrations section.

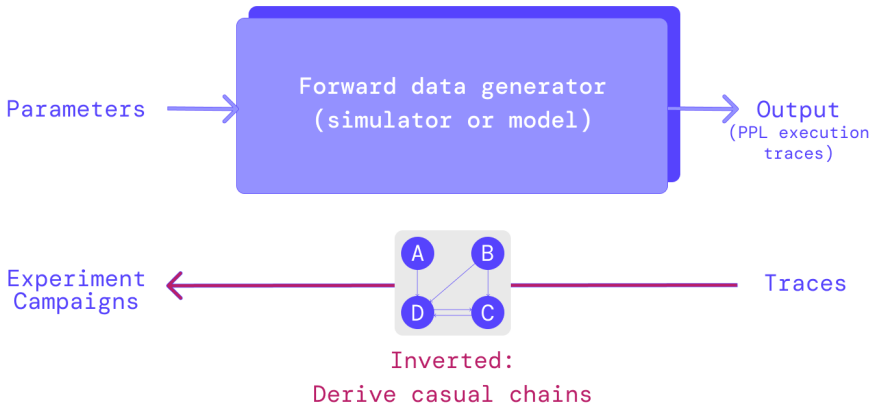

Figure 14: Inverting traditional forward data generation models – recall that a simulator aims to encode the cause-to-effect data generation process – to learn what causes could have produced the observed effects, and then to efficiently generate experimental campaigns to test these hypotheses. In practice, simulators that enable this cause-effect inversion learning can represent a new workflow that would broaden the scientific method [212].

Examples

Figure 15: Simulation runs from a virus-propagation agent based model (ABM), where red nodes are infected agents. With root-cause analysis based on the defined simulation-based claims of causality, the agents were partitioned into two distinct groups of independent causal chains (of infection). (Original figure from Ref. [211])

Tracing causal epidemiology chains with *in silico* interventions Systems of networked entities or agents that evolve over time can be difficult to understand in a practical way towards decision-making. For instance, the propagation of a virus in an environment, where it can be useful to identify "patient zero" or the agent that initiated the chain of infection. Methods for intervention-based counterfactual analysis can be used in agent-based modeling simulation to determine the root cause for each eventually infected agent.

First the researchers [211] begin with a simple counterfactual simulation experiment without intervention – that is, only manipulating the initial infection states before starting the simulation, not actively intervening in the simulation execution. This exemplifies the standard use of simulators for counterfactual reasoning: modify the input variables and environment parameters, observe the change in outputs per run, and assume cause-effect relationships. This inactive approach is susceptible to *preemption confounding*, where only manipulating the initial state of the disease transmission model may open the door for spurious background effects to creep in, and, from an observer's perspective, mask the original causal relationship. Thus, except for very simplistic simulations, inactive counterfactual reasoning is not necessarily reliable for determining root causes. To actively intervene in the simulation, one needs to implement a series of experiments with both negated infections and altered times of infection. In a small testbed of 10 agents, the root cause analysis produces a correct partitioning of the agents into two groups, where each group shared a common root cause infection, shown in Fig. 15. This demonstration is simplistic in the representation and dynamics of agent states. A more interesting use-case for future work would be simulations with multi-agent policies, such as the economic simulations we discuss next in the agent-based modeling motif, where one could actively intervene to discover why agents make certain decisions. Additional challenges to this approach can arise when considering multiple competing and interacting causal chains, which is to be expected in simulations of complex scenarios.

Active causal discovery and optimizing experiment design Asking the causal questions discussed above requires a causal model to begin with. The problem of inferring such a model from observational, interventional, or mixed data is called *causal discovery*. Dedicated causal discovery algorithms exist and can be separated into two subtypes: constraint-based and score-based. The constraint-based algorithms construct the causal structure based on conditional independence constraints, while the score-based algorithms generate a number of candidate causal graphs, assign a score to each, and select a final graph based on the scores [213]. For an overview of the causal discovery subfield see Glymour et al. [214]. Here we're interested in *active* causal discovery, illustrating a method that can be a novel "human-machine teaming" workflow.

The setting for active causal discovery involves an agent iteratively selecting experiments (or targeted interventions) that are maximally informative about the underlying causal structure of the system under study. The experiment process is shown in Fig. 16. At each iteration the agent,

1. Performs an experiment in the form of intervening on one of the variables and fixing its value, $do(X_j = x)$;
2. Observes the outcome by sampling from the interventional distribution $P(X_{-j}|do(X_j = x))$;
3. Updates its beliefs with the observed outcome;
4. Plans the next experiment to maximize information gain (or similar criteria).

The process repeats until the agent converges on a causal world model (with sufficient confidence) or an experiment threshold is reached. We use the syntax $do()$, which refers to Pearl's "*do*-calculus" [215, 216]. In short, the *do*-operator forces a variable to take a certain value or distribution, which is distinct from conditioning that variable on a value or distribution. This distinct operation allows us to estimate interventional distributions and causal effects from observational distributions (under certain conditions). Notice in the SI-stack Fig. 1 that *do*-calculus is stated explicitly with the causality module.

The active learning approach with targeted experiments allows us to uniquely recover the causal graph, whereas score- or constraint-based causal discovery from observational data can struggle going beyond the Markov equivalence class [217]. In the simulation setting we have virtual agents, while real scientists can follow the same procedures in the lab, potentially to minimize the experiments needed on costly equipment – notice the relationship to the sequential design-of-experiment methods we previously discussed, but now with the mathematics of causality. One can also imagine several variations involving human-machine teams to iterate over causal models and interventions (both *in silico* and *in situ*).

Von Kügelgen et al. [217] introduce a probabilistic approach with Bayesian nonparametrics, and there's been one application we know of, in the domain of neurodgenerative diseases [218]. The main developments are twofold:

1. The causal models contain continuous random variables with non-linear functional relationships, which are modeled with *Gaussian process (GP)* priors. A GP is a stochastic process which is fully specified by its mean function and covariance function such that any finite set of random variables have a joint Gaussian distribution

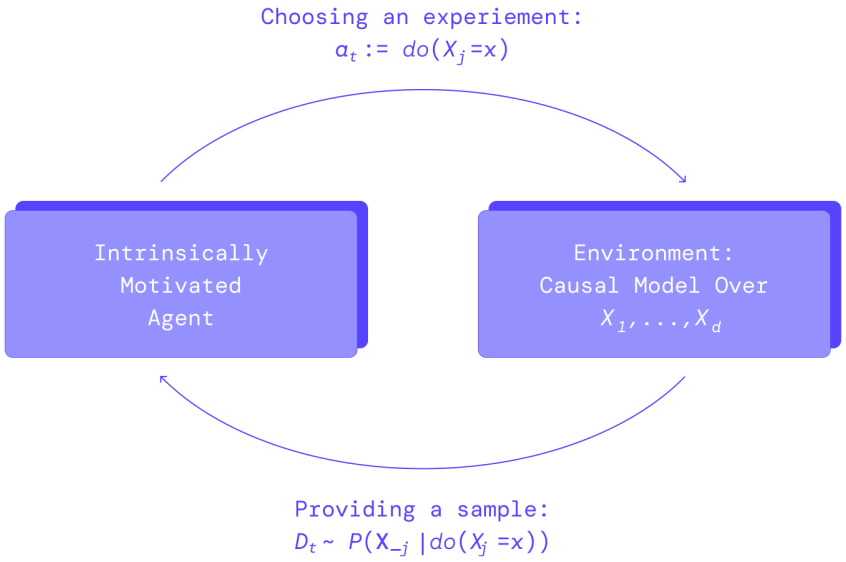

Figure 16: "Active" Bayesian causal discovery, where an *in silico* agent iteratively experiments with its simulation environment in order to learn a structured causal model. This simplified diagram shows a causal model learned over observed variables, where the agent actively intervenes on some of the variables to derive cause-effect relationships.

[219]. GPs provide a robust method for modeling non-linear functions in a Bayesian nonparametric framework; ordinarily one considers a GP prior over the function and combines it with a suitable likelihood to derive a posterior estimate for the function given data. GPs are flexible in that, unlike parametric counterparts, they adapt to the complexity of the data. GPs are largely advantageous in practice because they provide a principled way to reason about uncertainties [27].

2. The space of possible interventions is intractable, so *Bayesian optimization (BO)* is used to efficiently select the sequence of experiments that balances exploration and exploitation relative to gained information and optimizing the objective. BO is a probabilistic ML approach for derivative-free global optimization of a function f, where f is typically costly to evaluate (in compute, time, or interactions with people and environment). BO consists of two main components: a Bayesian statistical model for modeling the objective function (typically a GP, thus with uncertainty quantification), and an acquisition function for deciding where to sample next [80, 85]. The general idea is to trade off computation due to evaluating f many times with computation invested in selecting more promising candidate solutions x, where the space X can represent parameters of a real-world system, a machine learning model, an *in situ* experiment setup, and so on.

The use of probabilistic ML methods here enables us to maintain uncertainty estimates over both graph structures and their associated parameters during the causal discovery process, and potentially to propagate uncertainty values to downstream tasks. We are further investigating how to reliably quantify uncertainties with human-machine causal discovery. Another promising direction for work is to consider using physics-infused GP methods (discussed in the surrogate modeling motif) as the BO surrogate model – we explore this and related combinations of motifs in the Integrations section later.

Building a driving simulator for causal reasoning Robust and reliable simulation tools can help address the overarching challenge in counterfactual (and causal) reasoning: one never observes the true counterfactual outcome. In theory, the parameters of simulation environments can be systematically controlled, thereby enabling causal relationships to be established and confounders to be introduced by the researcher. In practice, simulators for complex environments are typically built on game engines [220] that do not necessarily suffice for causal reasoning, including the simulation testbeds for AI agents such as Unity AI [221] and OpenAI Gym [222]. We suggest several main limitations to overcome:

1. Counterfactual reasoning in general is not as simple as changing a variable and observing the change in outcomes. Rather, principled counterfactual reasoning is more precise: quantifying the potential outcomes under the Rubin-Neymann causal model [206] and within the constraints of potential confounders to derive

true cause-effect relationships. For example, the implementation of counterfactual Gaussian Processes [223] to simulate continuous-time potential outcomes of various clinical actions on patients.

2. *Confounders* – factors that impact both the intervention and the outcomes – can be known or measurable in some cases and hidden in others. The challenge here is to enable the researcher to systematically introduce, remove, and examine the impact of many different types of confounders (both known and hidden) while experimenting with simulations. As mentioned in the ABM example, only manipulating simulation inputs is subject to preemption confounding.
3. Prior work with causality-specific simulators has remained in relatively small environments with simplistic entities and actions, for example balls or objects moving on 2D and 3D surfaces [224, 225, 226].

The recent "CausalCity" [227] simulation environment aims to provide explicit mechanisms for causal discovery and reasoning – i.e., enabling researchers to create complex, counterfactual scenarios including different types of confounders with relatively little effort. The main novelty over similar simulation environments is the notion of agency: each entity in the simulation has the capability to "decide" their low-level behaviors, which enables scenarios to be designed with simple high-level configurations rather than specifying every single low-level action. They suggest this is crucial to creating simulation environments that reflect the nature and complexity of these types of temporal real-world reasoning tasks. CausalCity addresses problem (3) above with a city driving simulation of configurable, temporal driving scenarios and large parameter spaces. However, the main aims of a causal-first simulator are left unfulfilled, with only parts of problems (1) and (2) addressed: The notion of counterfactual reasoning in CausalCity is to forward simulate scenarios with specific conditions or parameters changed, and observe the difference in outcomes, meaning the counterfactual claims are limited to the full sequence of events rather than specific cause-effect relationships in the space between simulation inputs and outputs. For analyzing confounders, CausalCity provides tools to control *extraneous variables* – all variables which are not the independent variable but could affect the results of the experiment – which do not necessarily imply confounding, let alone show specific confounding relationships (for instance, as you would in an SCM).

Thus the CausalCity simulator does not include the means for deriving the precise causal mechanisms through direct interventions, nor control of causal inference bounds due to confounder assumptions. There needs to be tooling for causal modeling directly on the simulator execution trace, as we suggest later in probabilistic programming -based causal simulations. Nonetheless CausalCity represents a useful step in the direction of simulators for causal reasoning.

Future directions

Causal reasoning facilitates the simulation of possible futures and prediction of intervention outcomes, leading to more principled scientific discovery and decision-making. In several of the preceding examples we highlighted "active" processes with causal machine learning and simulators. Our belief is further development and validation can bring these active causal reasoning workflows into scientific practice to accelerate discovery and experimentation in diverse domains. Further, one can make the case that human-machine inference methods (Fig. 12) embodied with causal reasoning are necessary to make progress towards the *Nobel Turing Challenge* [228] to develop AI-scientists capable of autonomously carrying out research to make major scientific discoveries – it'd difficult to imagine such an AI-scientist without, for example, the ability to intelligently explore-exploit a system to derive its causal structure as in Fig. 16.

The specific directions of work we described are,

- Causality definitions and inference constraints based on simulation trace relationships
- Potential outcomes methods purpose-built for simulation counterfactual reasoning
- Tooling for active interventions on running simulations
- Tooling for autonomous and human-machine active causal discovery
- Game engines and testbeds with infrastructure and programmable interfaces for principled counterfactual (and interventional) reasoning

Another direction we propose in the Integrations section later is probabilistic programming as a formalism for causal reasoning, as a natural representation framework and also enabling probabilistic causal reasoning. Under the umbrella of advancing probabilistic causal reasoning, and continuing the thread from our causal optimization example above, work on causal inference with Bayesian optimization can offer new, improved approaches for decision making under uncertainty. Aglietti et al. [229] propose an intriguing approach for global causal optimization, and continuing this line work is an important direction – for example, combining their proposed framework with causal discovery algorithms, to remove dependency on manual encoding graphs while also accounting for uncertainty in the graph structure.

Perhaps the most important contribution of causal reasoning in simulation intelligence systems comes from the fact that correlations may well indicate causal relationships but can never strictly *prove* them. A model, no matter how robust, that reproduces the statistical properties of data, is a rather weak form of validation since there are many possible causal structures for a given set of statistical dependencies. Building simulators with proper representations and tooling for causal inference is critical for validating that simulators faithfully represent the true data generating processes, and ultimately for the reliability of simulation outcomes. One must consider the ethical implications of downstream actions and decision-making based on such outcomes, from epidemiology and medicine, to economics and government policy [122]. Simulation-based approaches can better deal with cause and effect than purely statistical methods from widely used in AI, ML, and data science, yet similar ethical concerns like model trust and interpretability [230], and algorithmic fairness [231, 232, 233] are crucial.

5. AGENT-BASED MODELING

In *agent-based modeling (ABM)* [234, 235], systems are modeled as collections of autonomous interacting entities (or agents) with encapsulated functionality that operate within a computational world. ABM agents can represent both individual or collective entities (such as organizations or groups). For example, in an artificial economy simulation for quantitatively exploring consequences of policy decisions, agents could be skilled and unskilled workers at varying class-levels in a city environment [236]. Or in biology, individual bacteria can be modeled as agents from population statistics obtained with flow cytometry [237].

ABM combines elements of game theory, complex systems, emergence, computational sociology, multi-agent systems, evolutionary programming, Monte Carlo methods, and reinforcement learning. Complex systems theory, for example, can be used to investigate how the interactions between parts can create collective behaviour within a dynamic system [238]. And the dynamic interaction between multiple agents or decision-makers, which simultaneously affects the state of the environment, can cause the emergence of specific and novel behavioral patterns, which can be studied with reinforcement learning techniques such as shaping reward structures and development of language between agents. The decision-making process with autonomous individuals in a bounded rational environment is often heterogeneous and lends itself well to ABM as a tool for analysis [239]; individuals represented by agents are dynamically interacting with other agents based on simple rules that will give rise to complex behaviours and patterns of displacement.

It is standard that agents can only acquire new data about their world constructively (through interactions), and agents can have *uncomputable beliefs* about their world that influence their interactions. Uncomputable beliefs can arise from inborn (initially configured) attributes, from communications received from other agents, and/or from the use of non-constructive methods (e.g., proof by contradiction) to interpret acquired data. These uncomputable beliefs enable agents to make creative leaps, i.e. to come up with new ideas about their world not currently supportable by measurements, observations, or logical extrapolations from existing information [240].

Formally a standard framework to use is *partial-observable multi-agent Markov Games (MGs)* [241] with a set of agents interacting in a state and action space that evolves over time—at each time step the environment evolves, and each agent receives an observation, executes an action, and receives a reward. Using machine learning to optimize agent behavior, each agent aims to learn a policy that maximizes its expected reward, which depends on the behavioral policies of the other agents and the environment transition dynamics. In a multi-agent economy setting, for example, the reward is modeled as a utility function, and rational economic agents optimize their total discounted utility over time. This describes the *individual perspective* in ABM, or aiming to achieve one's own goals. An additional *planner perspective* instead seeks to achieve some notion of social welfare or equilibrium for an interacting population, which usually involves intervening on population-level dynamics through policies, institutions, and centralized mechanisms [242]. We describe this in the context of a machine learned social planner to improve societal outcomes with the first example later.

The utility of ABM has proved challenging due to the complexity of realistically modeling human behavior and large environments (despite the ability of ABM to adopt behavioral rules that are deduced from human experiments [243]). Recent advances in deep reinforcement learning (RL) provide opportunity to overcome these challenges by framing ABM simulations as *multi-agent reinforcement learning (MARL)* problems. In MARL, agents need to learn together with other learning agents, creating a non-stationary environment [244]. The past few years have seen tremendous progress on games such as go [245, 246], StarCraft II [247], Dota2 [248], and two-player poker [249]. These represent *two-player zero-sum* games, which are convenient testbeds for multi-agent research because of their tractability: the predefined solutions coincide with the Nash equilibrium, the solutions are interchangeable, and they have worst-case guarantees [250]. Yet these testbeds limit research to domains that are inherently adversarial, while non-warfare

real-world scenarios such as simulated economies imply cooperative objectives and dynamics, where agents learn to communicate, collaborate, and interact with one another. The MARL field is starting to trend towards games of pure common interest, such as Hanabi [251] and Overcooked [252], as well as team games [253, 254]. *Mixed-motives* settings are more aptly connected to real-world settings, for example in alliance politics [242] and tax policy [10], which are important to research since some strategic dynamics only arise in these settings, such as issues of trust, deception, and commitment problems. The field of *cooperative AI* investigates a spectrum of these common-vs-conflicting interests dynamics, as well as myriad types of cooperative agents—machine-machine, human-machine, human-human, and more complex constellations. While much AI research to date has focused on improving the individual intelligence of agents and algorithms, cooperative AI aims to improve *social intelligence*: the ability of groups to effectively cooperate to solve the problems they face. Please see Dafoe et al. [242] for a thorough overview.

Reinforcement learning (including MARL and cooperative AI) provides a useful toolbox for studying and generating the variety of behaviors agents develop. The primary factor that influences the emergence of agent behavior is the reward structure, which can be categorized as cooperative, competitive, or intrinsic [255]. Intrinsic motivation implies the agent is maximizing an internal reinforcement signal (intrinsic reward) by actively discovering novel or surprising patterns, a concept which resembles *developmental learning* [256, 257] where organisms have to spontaneously explore their environment and acquire new skills [258, 259]. In addition to the various reward strategies, the development of language corpora and communication skills of autonomous agents is significant in emerging behaviors and patterns. Gronauer et al. [255] categorize this as *referential* and *dialogue*. The former describes cooperative games where the speaking agent communicates an objective via messages to another listening agent, and the latter includes negotiations, question-answering, and other back-and-forth dialogues between agents. Multi-agent communication can in general improve task performance and cumulative reward, but quantifying communication remains an open question – numerical, task-specific performance measurements provide evidence but do not give insights about the communication abilities learned by the agents.

It is natural for ABM testbeds to be implemented in game environments; while the AI Economist uses a simple 2-dimensional grid environment, the Dota2 and Starcraft II deep RL examples utilize those full video games. More generally, game engines such as Unity [221] provide powerful simulation environments for MARL and ABM problems broadly. Minecraft, for instance, has been the testbed for RL and human-AI interaction [260], and the "CausalCity" [227] simulation environment we discussed earlier is built with the Unreal game engine (leveraging the AirSim [261] package for simulations for autonomous vehicles). Game- and physics-engines are increasingly useful components of the AI toolbox and the SI ecosystem.

Examples

The ABM toolbox is applicable for study of complex interactive processes in all scientific disciplines. ABM is particularly interesting for simulating and studying systems in social sciences and economics, given the relative ease of mapping agents to recognizable social entities, the natural hierarchical self-organization in social systems, and the interesting results in emergent behavior – applications range from urban traffic simulation [262] to humanitarian assistance [263] to emergency evacuations [264]. Complex emergent economic behavior has been previously studied in economics through agent-based modeling [235, 265], but this has largely proceeded without the benefit of recent advances in AI. We illustrate AI-infused ABM towards societal intelligence with a couple of examples below, followed by additional domains in life sciences.

As with the other motifs, we advocate for probabilistic approaches to ABM, to properly calibrate models, assimilate observational data into models, and quantify uncertainties in results. For instance, we may make an agent's behavior stochastic in order to express our uncertainty in the agent's behavior, and we may implement Monte Carlo methods to compute a posterior of behaviors rather than single point predictions. In some cases we may be able to build a surrogate model of the ABM simulation (i.e. statistical emulation) for efficient execution and uncertainty quantification [266, 267].

Optimizing economic policies The "AI Economist" [10] aims to promote social welfare through the design of optimal tax policies in dynamic economies using ABM and deep RL. It uses a principled economic simulation with both workers and a policy maker (i.e. individual agents and a planner, respectively), all of whom are collectively learning in a dynamic inner-outer reinforcement learning loop (see Fig. 17). In the inner loop, RL agents gain experience by performing labor, receiving income, and paying taxes, and learn through trial-and-error how to adapt their behavior to maximize their utility. In the outer loop, the social planner adapts its tax policy to optimize the social objective–a social welfare function from economic theory that considers the trade-off between income equality and productivity. The AI Economist framework is designed in a way that allows the social planner to optimize taxes for any social objective,

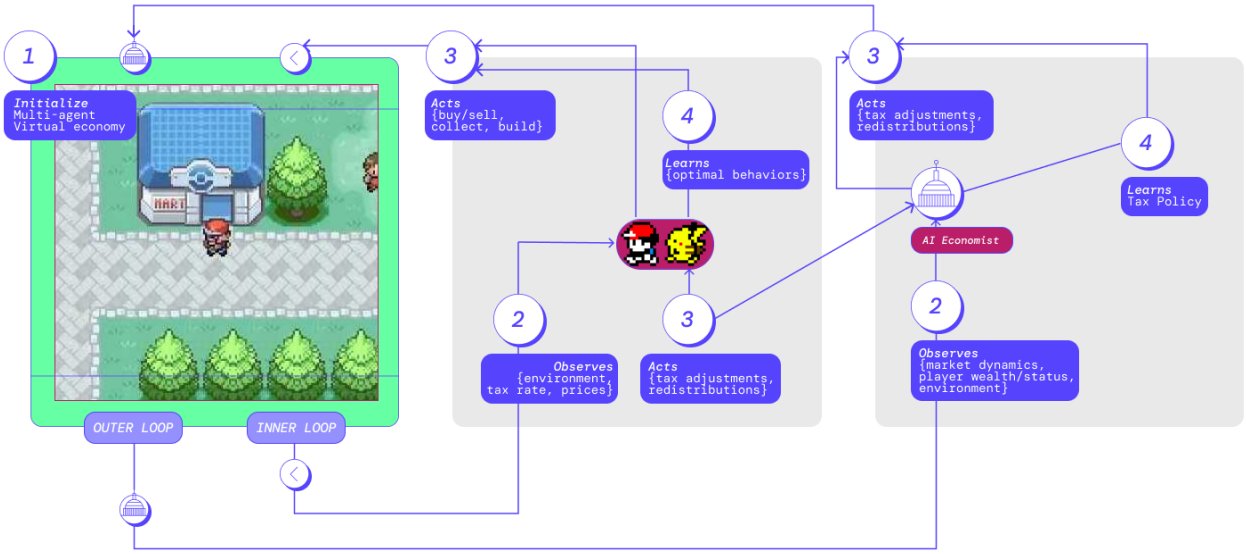

Figure 17: The AI Economist's two-level RL framework [10]. In the inner loop, various types of RL agents gain experience by performing labor, receiving income, and paying taxes, and learn through balancing exploration and exploitation how to adapt their behavior to maximize their utility. In the outer loop, the social planner adapts tax policies to optimize its social objective. The initial framework ran in a simplistic 2D environment with limited agency, whereas we show an example of work-in-progress extending AI Economist methods to more elaborate, complex multi-agent worlds [268].

which can be user-defined, or, in theory, learned. Another key benefit is the planner in this ABM setup does not need prior world knowledge, prior economic knowledge, or assumptions on the behavior of economic agents.

The data-driven AI Economist simulations are validated by matching the learned behavior to results known from economic theory, for example *agent specialization*. Specialization emerges because differently skilled workers learn to balance their income and effort. For example, in some scenarios agents learn tax-avoidance behaviors, modulating their incomes across tax periods. The dynamic tax schedule generated by the planner performs well despite this kind of strategic behavior. This exemplifies the rich, realistic dynamics and emergent behavior of a relatively simplified economic simulation. The validations also build confidence that agents respond to economic drivers, and demonstrate the value of a data-driven dynamic tax policy versus prior practice.

The AI Economist provides a powerful framework for integrating economics theory with data-driven simulation, producing transparent and objective insights into the economic consequences of different tax policies. Yet there are limitations: the simulations do not yet model human-behavioral factors and interactions between people, including social considerations, and they consider a relatively small economy. It would also be interesting to improve the framework with counterfactual reasoning for *what-if* experimentation – this is a key value-add of AI-driven simulators in general, which we detail in the causality motif.

Simulating networked acts of violence Particularly intriguing use-cases for new ABM approaches are in seemingly random and chaotic human events, such as insurrections and other acts of violence. Political revolutions, for example, are not understood nearly well enough to predict: similar patterns lead to heterogeneous scenarios, the cascading effects of actions (and their timing) have high uncertainty with expensive consequences, origins of emergent behaviors are often unknown and thus unidentifiable in the future, and it is impossible to know which episodes lead to local versus mass mobilization [269]. Other ABM concerns are especially relevant in these applications: data limitations (particularly challenges with multi-modal, sparse, noisy, and biased data), as well as challenges validating simulations (and further, trust).

Recent work applying ABM methods to violence events, such as aiming to model the insurrection events of the Arab Spring [269], can yield interesting outcomes, but nonetheless suffer from critical roadblocks: existing approaches use simplifying assumptions without real-world validation, cannot incorporate real data (historical nor new), and are not necessarily interpretable nor probe-able. Another is the agent-based insurgency model of Overbey et al. [270], where they simulate hundreds of thousands of agents that can exhibit a variety of cognitive and behavioral states and actions, in an environment loosely based on the demographic and geophysical characteristics of Ramadi, Iraq in the

early 2000s. Large scale parallel compute with GPUs was critical to produce simulations of realistic insurgent and counter-insurgent activities. Yet more acceleration and scaling would be needed for validation, including generating a posterior of potential outcomes for quantifying uncertainty and reasoning about counterfactual scenarios. These two use-cases exemplify the bottleneck ABM faces in geopolitical settings without newer AI-driven simulation techniques.

One important development would be multi-scale formalisms that are amenable to reinforcement learning and machine learning in general. A nice example in this direction is the hierarchical multi-agent, multi-scale setup of Bae & Moon [271], which was effective in simulating evacuation through a city's road networks due to bombing attacks, and simulating ambulance effectiveness transporting victims. For overcoming compute bottlenecks, existing HPC systems can be leveraged, but NN-based surrogates would be needed, for acceleration and additional emulation uses (as described in the emulation motif earlier).

Existing works in geopolitics ABM rely solely on mechanistic, theoretical descriptions of human behavior rooted in theory [272]. And recent work with deep RL uses simplistic board-game environments [273]. For reliable simulations it's important to utilize real-world data in a way that complements and validates the theory. In the applications here, we would implement a semi-mechanistic modeling approach that coordinates both mechanistic equations (such as information theoretic models of disinformation [274]) and data-driven function approximations. Further, building this ABM framework in a generative modeling (and more specifically probabilistic programming) way would be essential for simulating the multitudes of outcomes and their full posteriors. The probabilistic programming approach would also behoove counterfactual reasoning, or quantifying the potential outcomes due to various courses of action, as discussed in both the probabilistic programming and causality motifs.

Multi-scale systems biology Currently a main challenge in molecular modeling is the complex relationship between spatial resolution and time scale, where increasing one limits the other. There are macroscopic methods that make simplifying assumptions to facilitate modeling of the target molecular system (such as ODEs and PDEs operating at the population-level), and microscopic methods that are much finer in spatial resolution but highly computationally expensive at temporal scales of interest (such as Brownian and molecular dynamics methods). The reader may recall a similar spatial-temporal cascade of simulation methods in the motif on multi-scale modeling (Fig. 5).

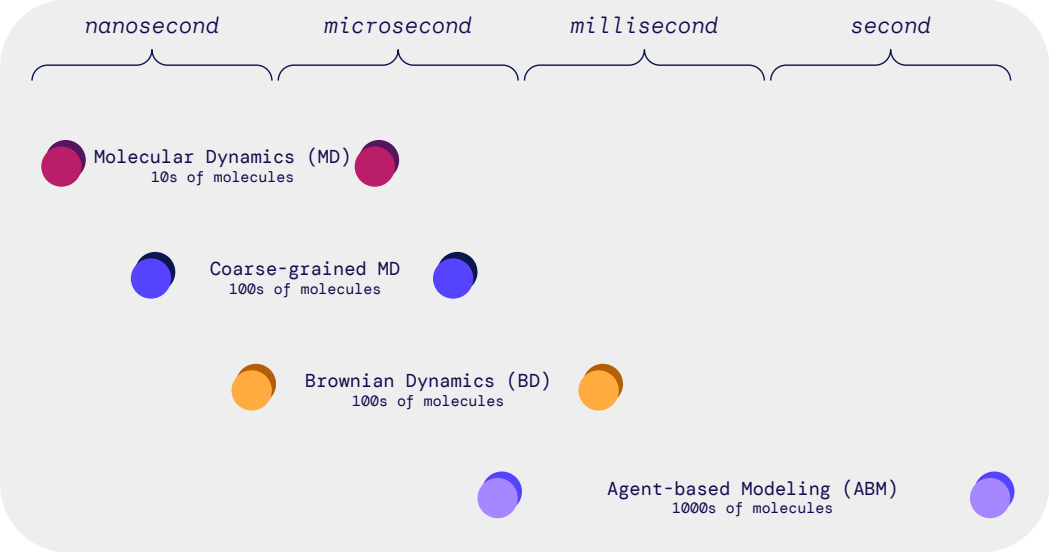

Figure 18: Comparing the size of the molecular system of interest and the timescales achievable by computational modeling techniques (motivated by [275]).

To highlight this cascade and how ABM can help overcome existing spatial-temporal systems biology limitations, consider several of the potentially many use-cases for modeling mRNA [275]:

- The nuclear export of mRNA transcripts requires a millisecond time scale, which has been explored via ABM [276], while high-resolution techniques such as Brownian or molecular dynamics are only tractable at nano- and micro-second scales, respectively. Further, these methods can compute 10s to 100s of molecules, whereas ABM can do 1000s.

- Cellular processes are often defined by spatial details and constrained environments – e.g., RNA-binding proteins can travel between the nucleus and the cytoplasm (depending on their binding partners), while others like the nucleo- and cyto-skeleton linker is within the nuclear envelope. These spatial characteristics can be easily incorporated in ABM as prior knowledge that improves modeling (Fig. 19).
- Tracking of individual particles over time is valuable in study of molecular systems, such as mRNA export. ABM is well-suited for particle tracking – for instance, recall the virus-propagation example in Fig. 15 of the causality motif – and can complement *in vivo* particle tracking advances [277] towards cyber-physical capabilities.
- The concept of emergence in ABM is valuable here as well – for instance, Soheileypour & Mofrad [278] use ABM to show that several low-level characteristics of mRNA (such as quality control mechanisms) directly influence the overall retention of molecules inside the nucleus. ABM, as a complex systems approach, has the ability to predict how a molecular system behaves given the rules that govern the behavior of individual molecules.

Another illuminating example is multi-scale ABM for individual cell signalling, cell population behavior, and extracellular environment. "PhysiBoSS" is an example framework to explore the effect of environmental and genetic alterations of individual cells at the population level, bridging the critical gap from single-cell genotype to single-cell phenotype and emergent multicellular behavior [279]. The ability of ABM to encompass multiple scales of biological processes as well as spatial relationships, expressible in an intuitive modeling paradigm, suggests that ABM is well suited for molecular modeling and translational systems biology broadly.

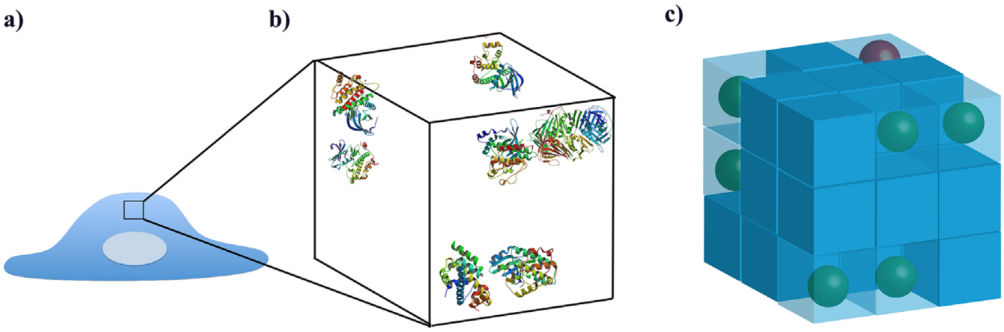

Figure 19: Simplification of representing a cell and its constituent objects (or agents) in a multi-scale agent-based model. (a) The cell system is characterized by various complex molecular pathways inside the nucleus (grey) and cytoplasm (blue), where each pathway takes place in particular environments and involves different types of molecules. The interactions of these molecules within and between cellular environments lead to the complex behaviors of the overall system. (b) A zoomed-in view of seven proteins interacting in an imaginary molecular environment in the cytoplasm. (c) ABM representation of the imaginary pathway shown in (b). Information about the environment is projected onto discrete ABM cells. Agents representing biological factors move and interact with other agents and the environment based on rules and mechanisms that are predefined (or potentially learned from data). Agents interact with other agents only when they are in proximity (i.e., green agents as opposed to the red agent). (Reproduced from [275])

Generating agent-based model programs from scratch In recent work, a combination of agent-based modeling and program synthesis via genetic programming was used to develop a codified rule-set for the behavior of individual agents, with the loss function dependent on the behaviors of agents within the system as a whole [280]. *Genetic programming* is the process of evolving computer programs from scratch or initial conditions: a set of primitives and operators, often in a domain-specific language. In *program synthesis*, the desired program behavior is specified but the (complete) implementation is not: the synthesis tool determines how to produce an executable implementation. Similarly, the behavior or any partial implementation can be ambiguously specified, and iteration with the programmer, data-driven techniques, or both are used to construct a valid solution.

Together, program synthesis via genetic programming allow non-trivial programs to be generated from example data. ABM is an intriguing area to apply this methodology, where emergence and individual-to-macro complexity are main characteristics to study and exploit. Focusing on two use cases – flocking behavior and opinion dynamics (or how opinions on a topic spread amongst a population of agents) – the researchers were able to generate rules for each agent within the overall system and then evolve them to fit the overall desired behavior of the system. This kind of approach

is early but has implications for large systems, particularly in sociological or ecological modeling. This and related approaches are emblematic of the artificial life field (ALife), which we discuss more in the open-ended optimization motif next.

Future directions

The previous examples elucidate a central feature of ABM which should help motivate the application of ABM in diverse domains: they are able to generate system-level behaviors that could not have been reasonably inferred from, and often may be counterintuitive to, the underlying rules of the agents and environments alone. The implication is there can be simulation-based discovery of emergent phenomena in many systems at all scales, from particle physics and quantum chemistry, to energy and transportation systems, to economics and healthcare, to climate and cosmology.

There are of course contexts in which the use of ABM warrants thorough ethical considerations, for instance ABM of social systems. In some applications, decision-making and social norms are modeled and inferred (e.g. of functional groups such as households, religious congregations, co-ops, and local governments [281]), which may unintentionally have elements of bias or ignorance when bridging individual-level assumptions and population-level dynamics [282]. There should be awareness and explicit systems in place to monitor these concerns in the use of ABM *and* for downstream tasks because the resulting predictions can inform the design or evaluation of interventions including policy choices.

Broadly in ABM, domain-specific approaches and narrow solutions are the norm. The lack of generalizable methods and unifying frameworks for comparing methods poses a significant bottleneck in research progress. Even more, this limits the utility of ABM for the many researchers interested in interdisciplinary simulations. A basic infrastructure of ABM is needed, with formalisms built on ABM logic and deep RL techniques (to integrate with state-of-art machine learning), and supporting the generic design of simulation engines working with multi-modal data. Building atop differentiable and probabilistic programming (the engine motifs discussed next) would behoove such an ABM framework. This is also a hardware problem: in line with the layer-by-layer integrated nature of simulation and AI computing that we describe throughout this paper, existing ABM simulation platforms for the most part can't take advantage of modern hardware, nor are they scalable to high-performance computing (HPC) deployments (which we discuss later in this work).

The AI Economist and similar works are moving towards an integration of traditional ABM methodology with newer deep RL approaches. In addition to powerful results, this can be a fruitful paradigm for studying language and communication between agents in many environments and settings – as mentioned earlier, the development of metrics and quantitative tools for studying such communication (and its emergent behaviors) is essential. It will likely be necessary to utilize the machinery of causal inference and counterfactual reasoning, for studying various communication types as well as motivations (e.g. [283]). It will also be critical to build methods in a hierarchical way for multi-scale ABM, to help overcome the burden of simplifying assumptions and approximations about individuals and groups of agents – notice the close similarity in challenges and methods described in the multi-scale multi-physics motif.

The Engine

6. PROBABILISTIC PROGRAMMING

The *probabilistic programming (PP)* paradigm equates probabilistic generative models with executable programs [284]. Probabilistic programming languages (PPL) enable practitioners to leverage the power of programming languages to create rich and complex models, while relying on a built-in inference backend to operate on any model written in the language. This decoupling is a powerful abstraction because it allows practitioners to rapidly iterate over models and experiments, in prototyping through deployment. The programs are generative models: they relate unobservable causes to observable data, to simulate how we believe data is created in the real world (Fig. 20).

PPL pack additional advantages for scientific ML, mainly by design allowing experts to incorporate domain knowledge into models and to export knowledge from learned models. First, PPL are often high-level programming languages where scientists can readily encode domain knowledge as modeling structure – there is a direct correspondence between probability theory and coding machinery that is obfuscated in standard programming paradigms. Second, since models are encoded as programs, program structure gives us a structured framework within which to reason about the properties and relationships involved in a domain before inference is even attempted [285]. Crucially, in PPL we get uncertainty reasoning as a standard feature allowing us to express and work with *aleatoric* (irreducible uncertainty due

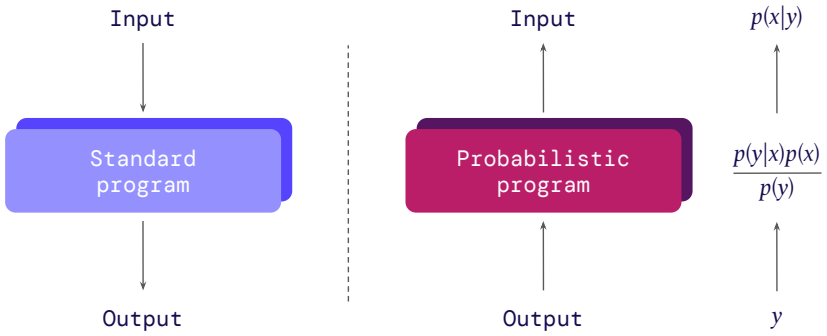

Figure 20: With a probabilistic program we define a joint distribution $p(x, y)$ of latent (unobserved) variables x and observable variables y. PPL inference engines produce posterior distributions over unobserved variables given the observed variables or data, $p(x|y) = p(y|x)p(x)/p(y)$. As observed variables typically correspond to the output of a program, probabilistic programming provides us a way to "invert" a given program, meaning that we infer program's inputs given an instance of data that corresponds to the program's output. Example PPL code is shown in Fig. 21. (Original figure credit: Frank Wood)

to stochasticity such as observation and process noise) and *epistemic* (subjective uncertainty due to limited knowledge) uncertainty measures.

The main drawback of probabilistic programming is computational cost, which is more pronounced especially in the case of unrestricted (universal, or Turing-complete) PPL [286] that can operate on arbitrary programs [287, 288, 289, 290, 291]. Another class of PPL is more restrictive and thus computationally efficient, where constraining the set of expressible models ensures that particular inference algorithms can be efficiently applied [292, 293, 294, 295]. Note that extending an existing Turing complete programming language with operations for sampling and conditioning results in a universal PPL [286, 296].

Although a relatively nascent field at the intersection of AI, statistics, and programming languages, probabilistic programming has delivered successful applications across biological and physical sciences. For example, inferring physics from high-energy particle collisions [164], pharmacometric modeling and drug development [297], inferring cell signalling pathways [298], predicting neurodegeneration [218], optimal design of scientific experiments [299], modeling epidemiological disease spread [300, 301], untangling dynamics of microbial communities [302], wildfire spread prediction [303], spacecraft collision avoidance [304], and more. We next discuss several key mechanisms for connecting probabilistic programming and scientific simulation across these and other domains.

Probabilistic programs as simulators First, *a probabilistic program is itself a simulator*, as it expresses a stochastic generative model of data; that is, given a system or process sufficiently described in a PPL, the forward execution of the program produces simulated data. Typically one runs a simulator forward to predict the future, yet with a probabilistic program we can additionally infer the parameters given the outcomes that are observed. For example, the neurodegeneration programs in Lavin [218] define a statistical model with random samplings, from which we can generate probabilistic disease trajectories, and infer parameters for individual biomarkers. Another advantage of simulating with probabilistic programs is for semi-mechanistic modeling, where Turing-complete PPL provide flexible modeling for integrating physical laws or mechanistic equations describing a system with data-driven components that are conditioned or trained on data observations – the PPL Turing.jl [305], for example, illustrates this with ordinary differential equations. Similarly, some PPL enable semi-parametric modeling [306, 307], where the inference engine can handle programs that combine parametric and nonparametric models: The parametric component learns a fixed number of parameters and allows the user to specify domain knowledge, while a nonparametric model (e.g. a Gaussian process) learns an unbounded number of parameters that grows with the training data. These two programming features – semi-mechanistic and semi-parametric modeling – along with the overall expressiveness of PPL make them ideal for non-ML specialists to readily build and experiment with probabilistic reasoning algorithms. Utilizing probabilistic programs as simulators plays an important role in the simulation-based inference motif.

Compiling simulators to neural networks Second, we have *inference compilation* [308, 309], a method for using deep neural networks to amortize the cost of inference in universal PPL models. The transformation of a PPL inference problem into an amortized inference problem using neural networks is referred to as "compilation", based on the analogy

Delay Differential Equation (DDE)
of Lotka-Volterra Model

$$dx/dt = \alpha x(t-\tau) - \beta y(t)x(t)$$

$$dy/dt = -\gamma y(t) + \delta x(t)y(t)$$

```
function delay_lotka_volterra(du, u, h, p, t)
   x, y = u
   α, β, γ, δ = p
   du[1] = α * h(p, t-1; idxs=1) - β * x * y
   du[2] = -γ * y + δ * x * y
   return
end

p = (1.5,1.0,3.0,1.0)
u0 = [1.0; 1.0]
tspan = (0.0,10.0)
h(p, t; idxs::Int) = 1.0
prob1 =
DDEProblem(delay_lotka_volterra,u0,h,tspan,p)
```

```
Turing.setadbackend(:forwarddiff)
@model function fitlv(data, prob1)

    σ ~ InverseGamma(2, 3)
    α ~ Truncated(Normal(1.5,0.5),0.5,2.5)
    β ~ Truncated(Normal(1.2,0.5),0,2)
    γ ~ Truncated(Normal(3.0,0.5),1,4)
    δ ~ Truncated(Normal(1.0,0.5),0,2)

    p = [α,β,γ,δ]

    prob = remake(prob1, p=p)
    predicted = solve(prob,saveat=0.1)
    for i = 1:length(predicted)
        data[:,i] ~ MvNormal(predicted[i], σ)
    end
end;
model = fitlv(ddedata, prob1)


# Draw samples using multithreading (3 independent chains
in parallel)
chain = sample(
    model, NUTS(.65), MCMCThreads(), 300, 3, progress=true)
```

Figure 21: Bayesian estimation of differential equations – Here we show an example of a Turing.jl [305] probabilistic program and inference with a *delay differential equation (DDE)*, a DE system where derivatives are functions of values at an earlier point in time, generally useful to model a delayed effect, such as incubation time of a virus. Here we show the The Lotka–Volterra equations, a pair of first-order nonlinear differential equations that are frequently used to describe the dynamics of biological systems in which two species interact as predator and prey. Differential equation models often have non-measurable parameters. The popular "forward-problem" of simulation consists of solving the differential equations for a given set of parameters, the "inverse problem" to simulation, known as parameter estimation, is the process of utilizing data to determine these model parameters. Bayesian inference provides a robust approach to parameter estimation with quantified uncertainty. (Example code sourced from turing.ml)

of compiling computer code between two programming languages. In the context of simulation, one use for inference compilation is as a preprocessing step for probabilistic programming algorithms, where initial runs of the PPL-based simulator are used to train a neural network architecture, which constructs proposal distributions for all the latent random variables in the program based on the observed variables – this represents one of the simulation-based inference workflows in Fig. 10. Another main use for inference compilation is to construct probabilistic surrogate networks (PSN): any existing stochastic simulator, written in any programming language, can be turned into a probabilistic program by adding a small amount of annotations to the source code (for recording and controlling simulator execution traces at the points of stochasticity). Munk et al. [296] demonstrate this on a non-trivial use-case with composite materials simulation, yet more testing with additional simulators is needed to validate the generalizability claims and robustness. PSN and other extensions to inference compilation [310] help unify probabilistic programming with existing, non-differentiable scientific simulators in the third and perhaps most powerful mechanism, simulator inversion.

Simulator inversion The *Etalumis* project ("simulate" spelled backwards) uses probabilistic programming methods to invert existing, large-scale simulators via Bayesian inference [164], requiring minimal modification of a given simulator's source code. Many simulators model the forward evolution of a system (coinciding with the arrow of time), such as the interaction of elementary particles, diffusion of gasses, folding of proteins, or the evolution of the universe on the largest scale. The task of inference refers to finding initial conditions or global parameters of such systems that can lead to some observed data representing the final outcome of a simulation. In probabilistic programming, this inference task is performed by defining prior distributions over any latent quantities of interest, and obtaining posterior distributions over these latent quantities conditioned on observed outcomes (for example, experimental data) using Bayes rule. This process, in effect, corresponds to *inverting the simulator* such that we go from the outcomes towards the inputs that caused the outcomes. A special protocol, called the probabilistic programming execution protocol (PPX) (Fig. 22), is used to interface PPL inference engines with existing stochastic simulator code bases in two main ways:

1. Record execution traces of the simulator as a sequence of sampling and conditioning operations in the PPL. The execution traces can be used for inspecting the probabilistic model implemented by the simulator, computing likelihoods, learning surrogate models, and generating training data for inference compilation NNs – see Fig. 22.

2. Control the simulator execution with the PPL by "hijacking" the simulator's random number generator. More precisely, general-purpose PPL inference guides the simulator by making random number draws not from the prior $p(x)$ but from proposal distributions $q(x|y)$ that depend on observed data y [311] – see Fig. 23.

These methods have been demonstrated on massive, legacy simulator code bases, and scaled to high-performance computing (HPC) platforms for handling multi-TB data and supercomputer-scale distributed training and inference. The ability to scale probabilistic inference to large-scale simulators is of fundamental importance to the field of probabilistic programming and the wider scientific community.

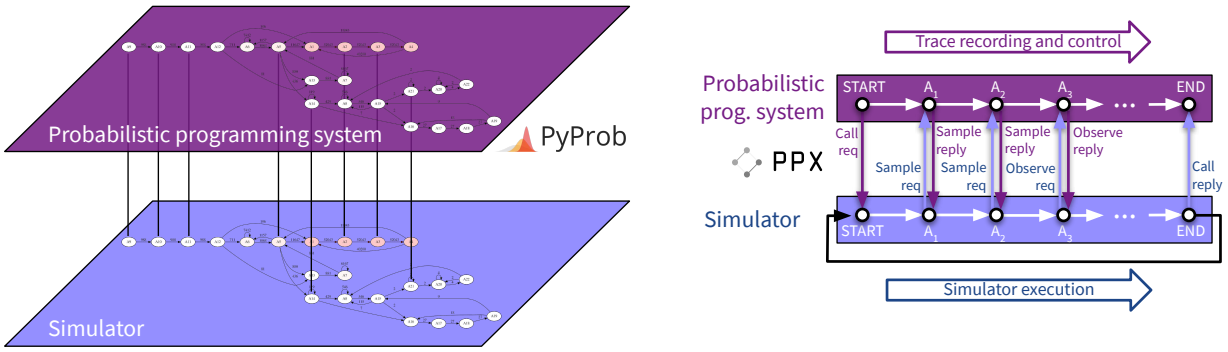

Figure 22: Illustration of the Etalumis probabilistic execution protocol (PPX) that interfaces the PPS (dark top-layer) with an existing simulator (light bottom-layer). As the simulator executes, the PPS inference engine records and controls the execution, as shown by the traces of nodes (representing unique sampled or observed variable labels at runtime). The two traces interface at random number draws and for conditioning in the simulator, corresponding to "sample" and "observe" statements in a typical PPL, respectively.

Examples

Compilation and inversion of a particle physics simulator The Standard Model of particle physics has a number of parameters (e.g., particle masses) describing the way particles and fundamental forces act in the universe. In experiments such as those at the Large Hadron Collider (described in the simulation-based inference motif earlier), scientists generate and observe events resulting in cascades of particles interacting with very large instruments such as the ATLAS and CMS detectors. The Standard Model is generative in nature, and can be seen as describing the probability space of all events given the initial conditions and model parameters. Baydin et al. [162, 164] developed the Etalumis approach originally to work with the state-of-the-art Sherpa simulator [312], an implementation of the Standard Model in C++. Based on the PPX protocol, they worked with this very large scale simulator involving approximately one million lines of C++ code and more than 25k latent variables and performed the first instance of Bayesian inference in this setting, demonstrating the approach on the decay of the τ (tau) lepton. Due to the scale of this simulator, the work made use of the Cori supercomputer at Lawrence Berkeley National Lab, mainly for distributed data generation and training of inference compilation networks for inference in Sherpa.

Probabilistic, efficient, interpretable simulation of epidemics Epidemiology simulations have become a fundamental tool in the fight against the epidemics and pandemics of various infectious diseases like AIDS, malaria, and Covid [313]. However, these simulators have been mostly limited to forward sampling of epidemiology models with different sets of hand-picked inputs, without much attention given to the problem of inference. Even more, inefficient runtimes limit the usability for iterating over large spaces of scenarios and policy decisions. Motivated by the Covid-19 pandemic, Wood et al. [314] demonstrated how parts of the infectious disease-control and policy-making processes can be automated by formulating these as inference problems in epidemiological models. Using the Etalumis approach, Gram-Hansen et al. [311] developed a probabilistic programming system to turn existing population-based epidemiology simulators into probabilistic generative models (Fig 23). Applied to two state-of-the-art malaria simulators (namely EMOD [315] and OpenMalaria [316]) the system provides interpretable introspection, full Bayesian inference, uncertainty reasoning, and efficient, amortized inference. This is accomplished by "hijacking" the legacy simulator codebases by overriding their internal random number generators. Specifically, by replacing the existing low-level random number generator in a simulator with a call to a purpose-built universal PPL "controller", the PPL can then track and manipulate the stochasticity of the simulator as it runs. The PPL controller can then run a handful of novel tasks with the hijacked codebase: running inference by conditioning the values of certain data samples and manipulating

others, uncovering stochastic structure in the underlying simulation code, and using the execution traces from the PPL backend to automatically produce result summaries. The resulting execution traces can potentially be used for causal chain discovery (for instance, deriving the infection spread from host to host) as described in the causality motif.

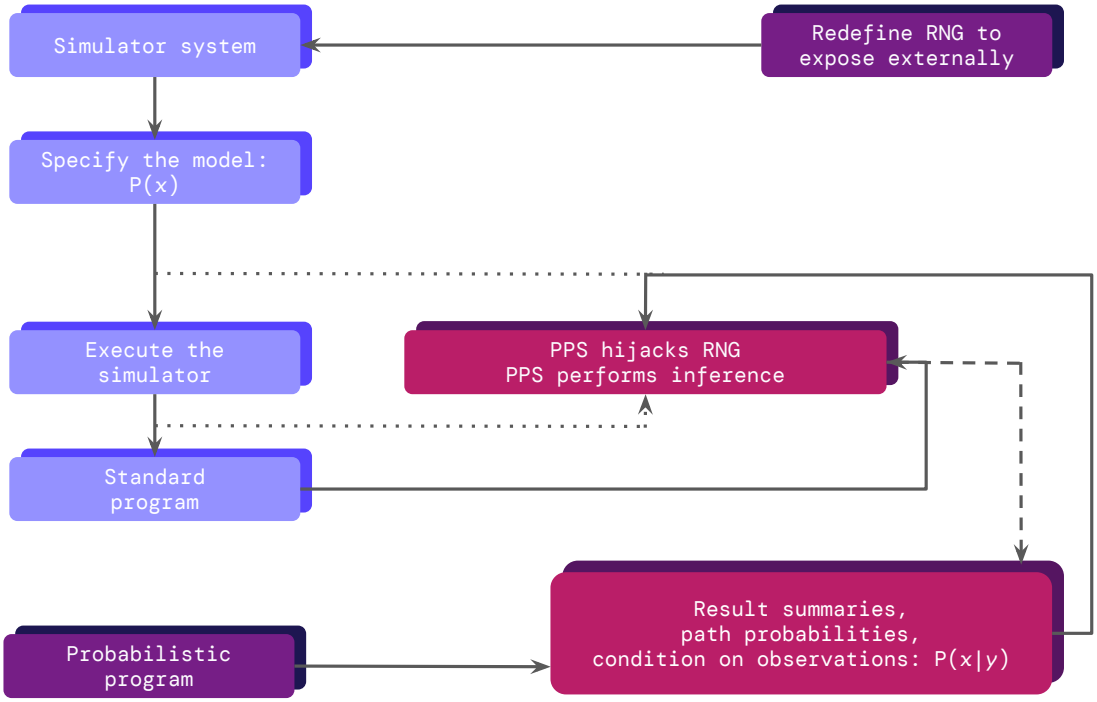

Figure 23: The process for "hijacking" the random number generator (RNG) of a generic population-based epidemiology simulator (such as OpenMalaria); the outline colors blue, red, and purple respectively represent processes linked to the existing simulator codebase, the hijacking probabilistic programming system, and external dataflows. (Reproduced from Gram-Hansen et al. [311])

The probabilistic programming advantage for generating molecules As we've covered in several motifs and integrations in this paper, designing and discovering new molecules is key for addressing some of humanity's most pressing problems, from developing new medicines to providing agrochemicals in the face of shifting populations and environments, and of course discovering new materials to tackle climate change. The molecular discovery process usually proceeds in design-make-test-analyze cycles (shown in Fig. 33). ML can help accelerate this process (ultimately reducing the number of expensive cycles) by both broadening the search and by searching more intelligently. That respectively amounts to learning strong generative models of molecules that can be used to sample novel molecules for downstream screening and scoring tasks, and finding molecules that optimize properties of interest (e.g. binding affinity, solubility, non-toxicity, etc.) [317]. We've suggested gaps in this process from lack of information sharing across phases, particularly the necessity for communication between the molecule design / lead generation phase and the synthesis phase – see for instance Fig. 37. One possibility to better link up the design and make steps in ML driven molecular generation is to explicitly include synthesis instructions in the design step. To this end, Bradshaw et al. [317] present a framework for multi-step molecular synthesis routes: directed acyclic graphs (DAG) represent is the synthetic routes, which are arranged in a hierarchical manner, where a novel neural message passing procedure exchanges information among multiple levels, all towards gradient-based optimization of the system that is enabled by differentiable programming. Intuitively this works as follows: The model aims to generate "recipes" (i.e., synthesis trees or synthesis graphs as DAGs) which start from a small number of building blocks (e.g., easily purchasable compounds) that then can be combined (possibly iteratively) to produce progressively more complex molecules. We assume that a "reaction outcome predictor" exists, which can be anything that takes in a set of reactants and outputs a distribution over products. Given a dataset of potential products and their synthetic routes, this can then be fit as structured deep generative model, which generates a distribution over molecules by first selecting a handful of easily obtainable compounds as starting points, then iteratively combining them. This generative model can then be used for Bayesian optimization of molecules (or perhaps open-ended optimization in SI future work), in a way which regularizes the "suggested" molecules to only be those that have a high probability of being synthesized reliably in a lab. We

discuss similar pipelines and methods in the Integrations section later. The overall structure of the probabilistic model is rather complex as it depends on a series of branching conditions. It's thus advantageous to frame the entire generative procedure as a probabilistic program – notice our location in reference to Fig. 25 at the intersection of DP and PP. Such a probabilistic program defines a distribution over DAG serializations, and running the program forward will sample from the generative process. The program can just as easily be used to evaluate the probability of a DAG of interest by instead accumulating the log probability of the sequence at each distribution encountered during the execution of the program. Inspecting the PPL execution traces may provide deeper levels of interpretability for the molecule generation process. This exemplifies the utility of probabilistic programs for "partially specified" models, which are the rich area between data-driven (black-box, deep learning) and fully-specified simulators. We give another example of this in the Integrations section later, also in the context of molecular synthesis.

Simulation-based common sense cognition The pursuit of common sense reasoning has largely been missing from mainstream AI research, notably characteristics such as generalizability and causal reasoning [318, 319]. To achieve common sense, and ultimately general machine intelligence, Marcus & Davis [320] suggest starting by developing systems that can represent the core frameworks of human knowledge: time, space, causality, basic knowledge of physical objects and their interactions, basic knowledge of humans and their interactions. Practical pursuit of this objective has largely focused on reverse engineering the "common sense core" from cognitive- and neuro-sciences: *intuitive physics* [321] and *intuitive psychology*, or basic representations of objects in the world and the physics by which they interact, as well as agents with their motives and actions [322]. In particular, studying the hypothesis that many intuitive physical inferences are based on a *mental physics engine* that is analogous in many ways to the *machine physics engines* used in building interactive video games [323], and by modeling the intuitive physics and intuitive psychology of children as mental simulation engines based on probabilistic programs [318, 324].

To capture these kinds of ideas in engineering terms probabilistic programming is needed; PPL integrate our best ideas on machine and biological intelligence across three different kinds of mathematics: symbolic manipulation (i.e., algebra and logic), Bayesian inference (probability), and neural networks (calculus). By wrapping physics engine programs and physics simulators inside frameworks for probabilistic modeling and inference, we gain the powerful tools necessary to capture common sense intuitive physics. In the works of Tenenbaum et al. this has been called "the simulator in your head", and also "theory of mind" [325], which has a purpose distinct from the typical use of game engines in most of recent AI research: to capture the mental models that are used for online inference and planning, rather than using games for training a NN or other model over a massive amount of simulated data.

These common sense cognition concepts have been demonstrated in a series of simulation-based cognitive science experiments to test and analyze the PPL-based intuitive physics engine [326, 327], and the intuitive psych engine [328] where the same kind of model can be used to infer jointly what somebody wants and also what somebody believes, (because people sometimes take inefficient actions [325]). Uncertainty reasoning is once again key here: the PPL-based approach leads to a more robust machine intelligence because any real-world AI agent will never have perfect perception nor perfect physics models, so, like the human brain, making probabilistic inferences based on the available data and models is a robust, generalizable approach. Follow-up neuroscience experiments with Tenenbaum et al. aim to show where and how your intuitive physics engine works, and how the neural mechanisms connect synergistically to other cognitive functions based on their neural locus. An implication is that these kinds of probabilistic program models are operative even in the minds and brains of young babies, which may bear significance in developing methods for *cooperative AI*, which we discuss in the agent-based modeling and uncertainty reasoning sections.

As we noted earlier, PPL sampling-based inference is predominant but in general slow, and certainly restrictive with games/graphics engines. We can alternatively use NNs to learn fast bottom-up approximations or inverses to these graphics engines, also known as "vision as inverse graphics" [329, 330]. Looking to lean on NNs, PhysNet [331] attempts to treat intuitive physics as a pattern recognition task. Although able to reproduce some of the stability-prediction results, typical of deep learning the PhysNet approach required massive amounts of training and still cannot generalize beyond the very specific physical object shapes and numbers.

Now a physics engine defined as a probabilistic program has to be a program that inputs and outputs other programs. This "hard problem of learning" represents a highly challenging search problem to find good programs, yet humans can solve that problem so machine intelligence should be able to as well. Bayesian Program Learning [332] is an early example of this but there is still considerable progress to be made. Additional notable works towards learning as programming include "the child as hacker" [333], and "Dream Coder" which combines neural networks with symbolic program search to learn to solve programs in many different domains, essentially learning to construct a deep library of programs [334].

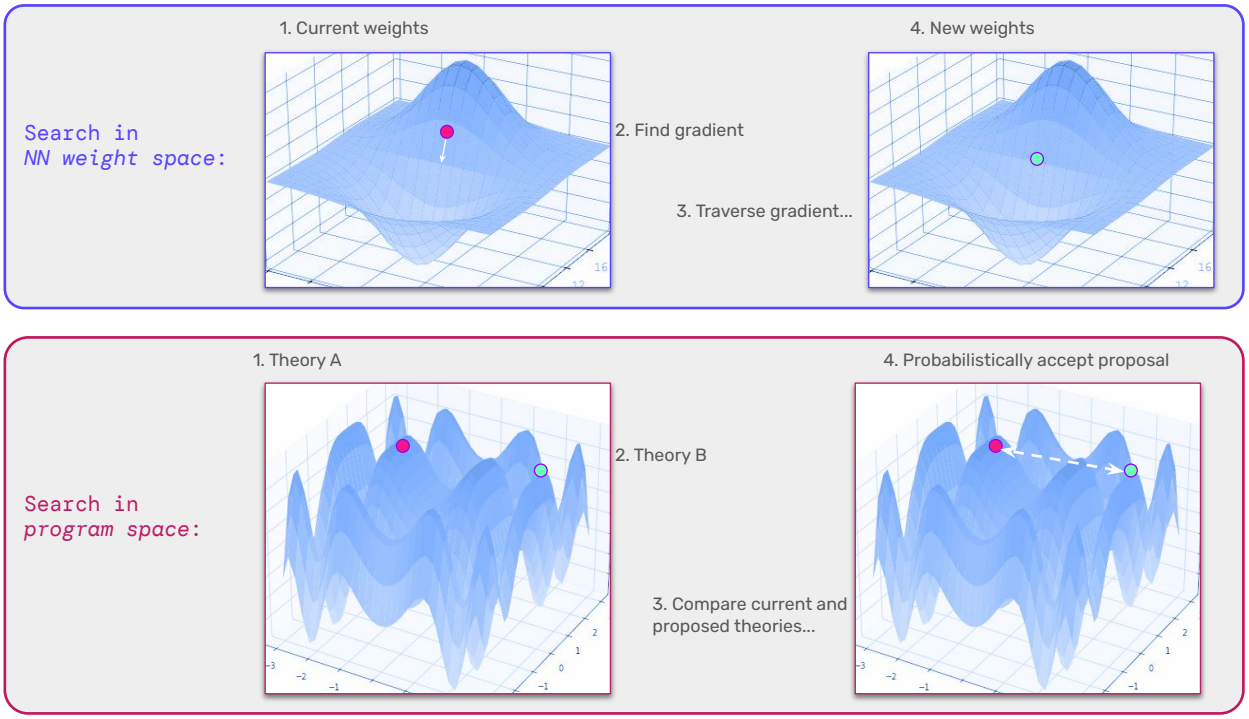

Figure 24: Search spaces reflecting the "hard problem of learning", where navigating the space of programs (bottom) is far more challenging than the space of neural net parameterizations (top). (Inspired by [324].)

Future directions

Probabilistic programming and simulation should continue to advance the cognitive- and neuro-sciences work we described above from Tenenbaum et al. The hypothesis, with a growing body of support, is that a simulator or game engine type of representation is what evolution has built into our brains in order to understand the world of objects and agents; babies start off with a minimal proto game engine, learn with the game engine and expand it to cases not necessarily built for, and use this as the subsequent foundation for learning everything else.

Another general area to advance and scale PPL use is in simulation-based inference. Specifically in the context of human-machine inference, using probabilistic programs as the simulator has many potential advantages: modularity in modeling systems, general-purpose inference engines, uncertainty quantification and propagation, and interpretability. See Fig. 12 and the simulation-based inference motif.

With the Etalumis project we now have the ability to control existing scientific simulators at scale and to use them as generative, interpretable models, which is highly valuable across simulation sciences where interpretability in model inference is critical. The proof of concept was developed in particle physics with the Large Hadron Collider's (LHC) "Sherpa" simulator at scale. This enables Bayesian inference on the full latent structure of the large numbers of collision events produced at particle physics accelerators such as the LHC, enabling deep interpretation of observed events. Work needs to continue across scientific domains to validate and generalize these methods, which has begun in epidemiology [301, 311] and astrophysics [304]. Other promising areas that are bottlenecked by existing simulators include hazard analysis in seismology [335], supernova shock waves in astrophysics [336], market movements in economics [337], blood flow in biology [49], and many more. Building a common PPL abstraction framework for different simulators would be ideal, for experiment-reproducibility and to further allow for easy and direct comparison between related or competing simulators.

In general, the intersection of probabilistic programming and scientific ML is an actively growing ecosystem. Still, much more progress stands to be made in computational efficiency, both in the inference algorithms and potentially in specialized compute architectures for PPL. We also believe the development of causal ML methods within PPL will be imperative, as the abstractions of probabilistic programming provide opportunity to integrate the mathematics of causality with ML software. The larger opportunity for probabilistic programming is to expand the scope of scientific modeling: Just as Turing machines are universal models of computation, probabilistic programs are universal

probabilistic models – their model space M comprises all computable probabilistic models, or models where the joint distribution over model variables and evidence is Turing-computable [324].

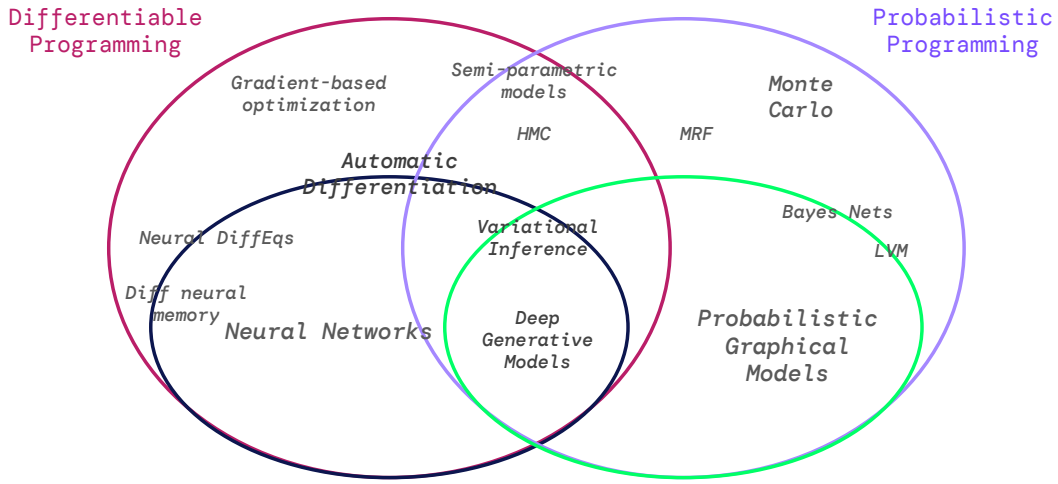

Figure 25: Probabilistic programming expands the scope of probabilistic graphical models (PGM) by allowing for inference over arbitrary models as programs, including programming constructs such as recursion and stochastic branching. Analogously, differentiable programming is a generalization of neural networks and deep learning, allowing for gradient-based optimization of arbitrarily parameterized program modules. Just as MCMC-based inference engines drive PPL, autodiff is the engine behind DP. (Note that "differentiable neural memory" refers to methods such as Neural Turing Machines [338] and Differentiable Neural Computers [339]; "neural differential equations" refers to Neural ODE [78] and Universal DiffEQ (UDE) [66]; HMC is Hamiltonian Monte Carlo [340, 341], MRF is Markov Random Fields, and LVM is Latent Variable Models [342, 343].)

7. DIFFERENTIABLE PROGRAMMING

Differentiable programming (DP) is a programming paradigm in which derivatives of a program are automatically computed and used in gradient-based optimization in order to tune program to achieve a given objective. DP has found use in a wide variety of areas, particularly scientific computing and ML.

From the ML perspective, DP describes what is arguably the most powerful concept in deep learning: parameterized software modules that can be trained with some form of gradient-based optimization. In the DP paradigm, neural networks are constructed as differentiable directed graphs assembled from functional blocks (such as feedforward, convolutional, and recurrent elements), and their parameters are learned using gradient-based optimization of an objective describing the model's task [344]. The ML field is increasingly embracing DP and freely composing NN building blocks in arbitrary algorithmic structures using control flow, as well as the introducing novel differentiable architectures such as the Neural Turing Machine [339] and differentiable versions of data structures such as stacks, queues, deques [345].

DP frameworks for deep learning have dominated the ML landscape during the past decade, most notably auto-differentiation frameworks such as Theano [346], TensorFlow [347], and PyTorch [348]. In contrast to deep learning, which mainly involves compositions of large matrix multiplications and nonlinear element-wise operations, physical simulation requires complex and customizable operations due to the intrinsic computational irregularities, which can lead to unsatisfactory performance in the aforementioned frameworks. To this end, recent works have produced powerful, parallelizable DP frameworks such as JAX [45, 349] and DiffTaichi [350] for end-to-end simulator differentiation (and thus gradient-based learning and optimizations), and each can integrate DP into simulator code. Both have promising applications in physical sciences, which we discuss more in the context of examples below. Further, the Julia language and ecosystem [351] provides system-wide differentiable programming, motivated by scientific and numerical computing, with recent AI integrations for deep learning and probabilistic programming.

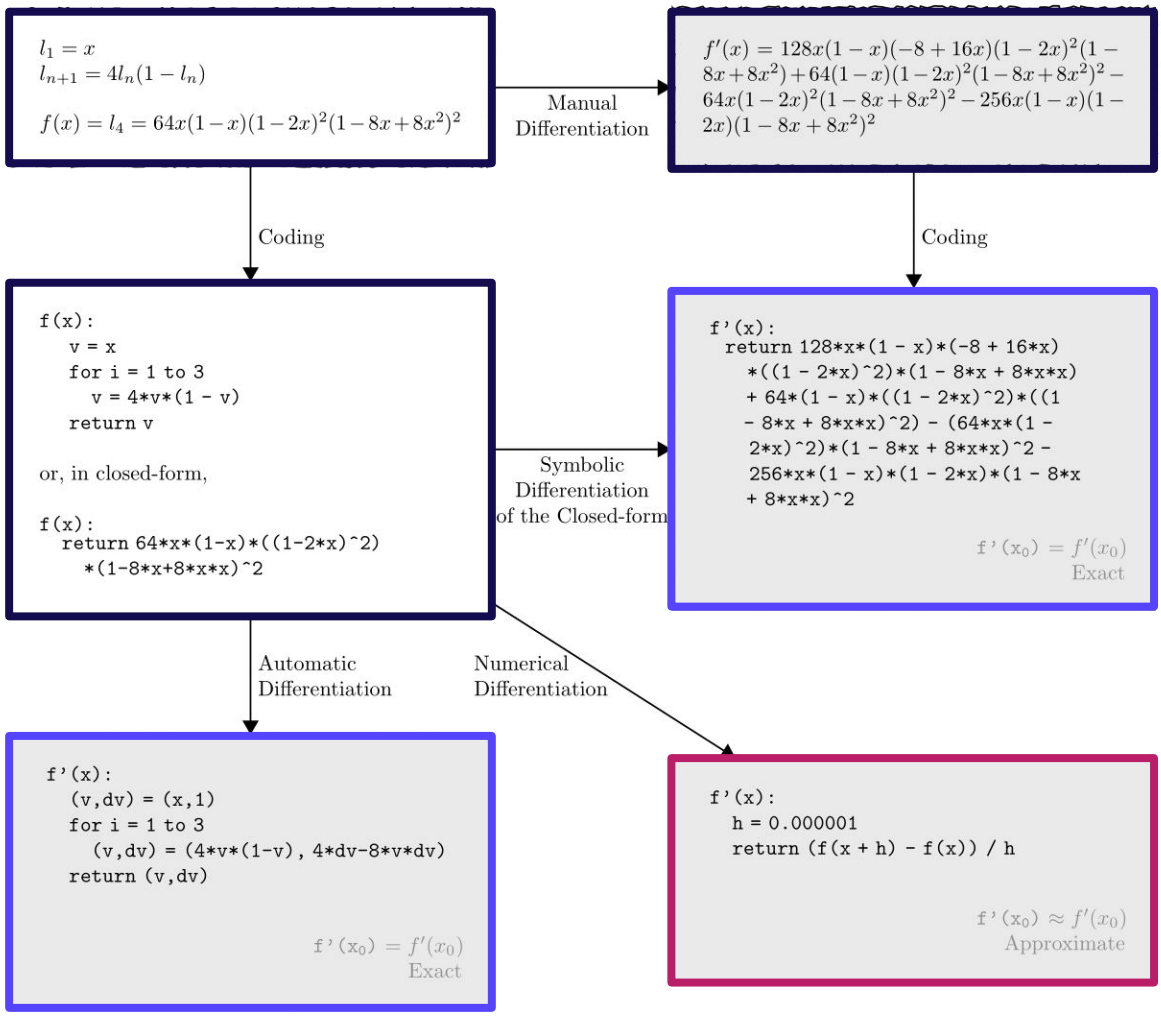

Figure 26: A taxonomy of differentiating mathematical expressions and computer code, taking as an example a truncated logistic map (upper left). Symbolic differentiation (center right) gives exact results but requires closed-form input and suffers from expression swell, numerical differentiation (lower right) has problems of accuracy due to round-off and truncation errors, and automatic differentiation (lower left) is as accurate as symbolic differentiation with only a constant factor of overhead and support for control flow. (Reproduced from Baydin et al. [344])

Automatic differentiation (AD) At the intersection of calculus and programming, AD is the workhorse of DP. AD involves augmenting the standard computation with the calculation of various derivatives automatically and efficiently [344]. More precisely, AD performs a non-standard interpretation of a given computer program by replacing the domain of the variables to incorporate derivative values and redefining the semantics of the operators to propagate derivatives per the chain rule of differential calculus. AD is not simply non-manual differentiation as the name suggests. Rather, AD as a technical term refers to a specific family of techniques that compute derivatives through accumulation of values during code execution to generate numerical derivative evaluations rather than derivative expressions. This allows accurate evaluation of derivatives at machine precision with only a small constant factor of overhead and ideal asymptotic efficiency. Availability of general-purpose AD greatly simplifies the DP-based implementation of ML architectures, simulator codes, and the composition of AI-driven simulation, enabling their expression as regular programs that rely on the differentiation infrastructure. See Baydin et al. [344] for a thorough overview.

In the context of simulators, DP, like probabilistic programming, offers ways to exploit advances in ML [1] and provides an infrastructure to incorporate these into existing scientific computing pipelines. An intriguing approach with many applications is machine-learned surrogate modeling with scientific simulators and solvers. In Fig. 27 we illustrate one example of chaining a neural network surrogate with a simulator (as an ODE solver) to solve an *inverse problem*, or inferring hidden states or parameters from observations or measurements. DP enables this approach because the

end-to-end differentiable workflow, including both simulation code and ML model, can backpropagate errors and make updates to model parameters. We discuss this and related methods in depth in the multi-physics and surrogate modeling motifs, and how all the motifs enable a spectrum of inverse problem solving capabilities.

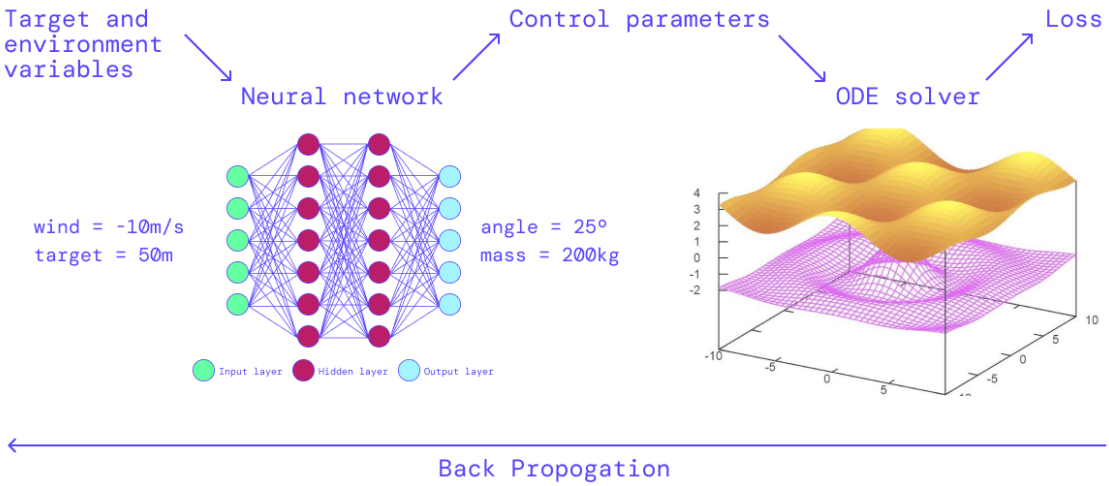

Figure 27: A neural network surrogate expressed in sequence with an ODE solver within a differentiable programming framework, which allows end-to-end learning by backpropagating the output loss through full system. This example illustrates an inverse problem: given the situation (wind, target) we need to solve for optimal control parameters (angle, weight) for the physical system (represented by an ODE-based simulator). (Figured inspired by Chris Rackauckas [352])

Examples

Differentiable pipeline for learning protein structure Computational approaches to protein folding not only seek to make structure determination faster and less costly (where protein structure is a fundamental need for rational drug design), but also aim to understand the folding process itself. Prior computational methods aim to build explicit sequence-to-structure maps that transform raw amino acid sequences into 3D structures [353], either with expensive molecular dynamics (MD) simulations or "fragment assembly" methods that rely on statistical sampling and a given template solution. Another category of prior methods are "co-evolution" where multiple sequence alignment is used to infer geometric couplings based on evolutionary couplings, yet in practice to go from the sequence contact map to a full 3D structure is only reliable 25%–50% of the time [354]. In general there are major shortcomings in the prior work such as inability to predict structural consequences from slight changes or mutations [355] and inability to work with *de novo* sequences [356], not to mention the computational burdens (for example, the prior state-of-art pipeline that combines template-based modeling and simulation-based sampling takes approximately 20 CPU-hours per structure [357]). To address the above limitations, AlQuraishi [356] proposes differentiable programming to machine-learn a model that replaces structure prediction pipelines with differentiable primitives. The approach, illustrated in Fig. 28, is based on four main ideas: (1) encoding protein sequences using a recurrent neural network, (2) parameterizing (local) protein structure by torsional angles to enable a model to reason over diverse conformations without violating their covalent chemistry, (3) coupling local protein structure to its global representation via recurrent geometric units, and (4) a differentiable loss function to capture deviations between predicted and experimental structures that is used to optimize the recurrent geometric network (RGN) parameters. Please see the original work for full details [356], including thorough experiments comparing quantitative and qualitative properties of the DP-based approach (RGN) versus prior art. One important highlight is the RGN approach is *6-7 orders of magnitude faster* than existing structure prediction pipelines, although there is the upfront cost of training RGN for weeks to months. Beyond results that overall surpass the prior methods (including performance on novel protein folds), the differentiable structure approach can be useful towards a bottom-up understanding of systems biology. AlphaFold, the recent breakthrough approach that surpasses RGN and everything else in the field [358], also exploits autodiff (Fig. 25 but in a deep-learned way that doesn't necessarily improve scientific knowledge of protein folding; more on AlphaFold is mentioned in the Honorable Mentions subsection later. A learning-based approach, in contrast to a deterministic pipeline, can provide valuable information towards future experiments and scientific knowledge. In this case of machine-learning protein sequences, the resulting model learns a low-dimensional representation of protein sequence space, which can then be explored, for example with search and optimization algorithms we've discussed in the open-endedness and multi-physics motifs.

In a standard pipeline (and to a different degree in AlphaFold), this knowledge is obfuscated or not present – the full domain knowledge must be encoded *a priori*. The use of simulation intelligence motifs in addition to DP holds great potential in this area, notably with semi-mechanistic modeling to incorporate experimental and structural data into integrative multi-scale simulation methods that can model a range of length and time scales involved in drug design and development [359].

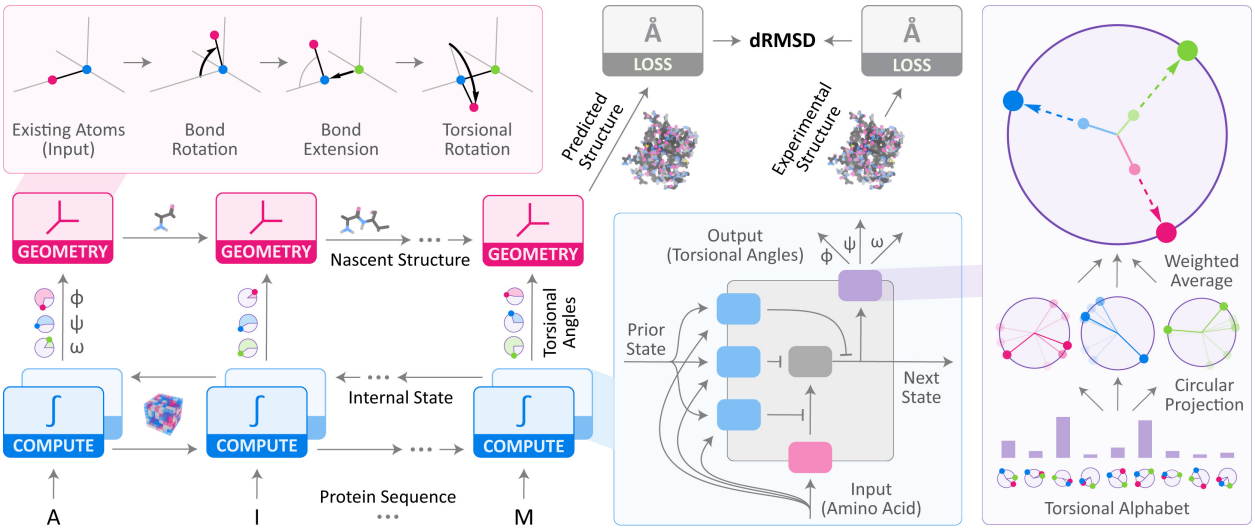

Figure 28: The recurrent geometric network (RGN) of [356]. The model takes a sequence of amino acids and position-specific scoring matrices as input and outputs a 3D structure by running three stages: computation (left, blue blocks), geometry (left, pink blocks), and assessment (right steps). (Original figure from Mohammed AlQuraishi [356])

JAX: DP for physical simulation from molecular dynamics to rigid body motion We've highlighted JAX several times now as a valuable system for utilizing machine learning to improve simulation sciences. One use-case demonstrating integration of several motifs is the example of building semi-mechanistic surrogate models in the DP framework in order to accelerate complex computational fluid dynamics (CFD) simulations [18]. Another use-case we alluded to in Fig, 5 is spanning multiple modeling scales including molecular dynamics (MD). MD is frequently used to simulate materials to observe how small scale interactions can give rise to complex large-scale phenomenology [45]. Standard MD packages used across physics simulations (such as HOOMD Blue [360] or LAMMPS [361, 362]) are complicated, highly specialized pieces of code. Poor abstractions and technical debt make them difficult to use, let alone extend for advances in accelerated computing and ML – for instance, specialized and duplicated code for running on CPU vs. GPU, hard-coded physical quantities, and complex manual differentiation code. For these reasons, JAX MD [50] leverages differentiable programming as well as automatic hardware acceleration (via the compilation scheme in JAX). Several design choices rooted in functional programming make JAX MD distinct from traditional physics software, and more capable with autodiff and ML methods: transforming arrays of data, immutable data, and first-class functions. JAX MD primitives are functions that help define physical simulations, such as "spaces" (dimensions, topology, boundary conditions) and "interactions" (energy, forces). Higher level functions can then be used to compose simulations of particles or atoms.

The initial JAX MD work [50] demonstrated training several state-of-art neural networks to model quantum mechanical energies and forces, specifically to represent silicon atoms in different crystalline phases based on density functional theory (DFT, the standard first principles approach to electronic structure problems). DFT is prevalent across physics problems but non-trivial to reliably compute or approximate in practice. In recent years there's been interest in utilizing ML to improve DFT approximations, notably towards JAX-DFT [363]. They leverage the differentiable programming utilities of JAX to integrate the DFT solver equations (i.e., Kohn-Sham equations) directly into neural network training – that is, leveraging JAX as a physics-informed ML framework. This integration of prior knowledge and machine-learned physics, the authors suggest, implies rethinking computational physics in this new paradigm: in principle, all heuristics in DFT calculations (e.g., initial guess, density update, preconditioning, basis sets) could be learned and optimized while maintaining rigorous physics [363]. JAX has similarly been leveraged for automated construction of equivariant layers in deep learning [364], for operating on graphs and point clouds in domains like particle physics and dynamical systems where symmetries and equivariance are fundamental to the generalization of neural networks – *equivariant neural networks* are detailed later in the Discussion section. A more AI-motivated extension of JAX is Brax

[365], a differentiable physics engine for accelerated rigid body simulation and reinforcement learning. Brax trains locomotion and dexterous manipulation policies with high efficiency by making extensive use of auto-vectorization, device parallelism, just-in-time compilation, and auto-differentiation primitives of JAX. For similar reasons, JAX has been leveraged specifically for *normalizing flows*, a general mechanism for defining expressive probability distributions by transforming a simple initial density into a more complex one by applying a sequence of invertible transformations [366, 367]. We touch on normalizing flows more in the Discussion section later.

DiffTaichi: purpose-built DP for highly-efficient physical simulation Taichi [368] is a performance oriented programming language for writing simulations, and DiffTaichi [350] augments it with additional automatic differentiation capabilities. While JAX MD and the various others in the JAX ecosystem can provide physical simulation workflows out of the box because they're purpose built for those applications, DiffTaichi is "closer to the metal" for fast simulations and thus in general better computational performance over JAX. This can be explained by several key language features required by physical simulation, yet often missing in existing differentiable programming tools:

- *Megakernels* – The programmer can naturally fuse multiple stages of computation into a single kernel, which is later differentiated using source code transformations and just-in-time compilation. Compared to mainstream DP frameworks for deep learning (PyTorch [348] and TensorFlow [347]) which use linear algebra operators, the megakernels have higher arithmetic intensity and are thus more efficient for physical simulation tasks.
- *Imperative parallel programming* – DiffTaichi is built in an imperative way for better interfacing with physical simulations than the functional array programming approach of mainstream deep learning frameworks. First, this approach helps provide parallel loops, "if" statements and other control flows in a differentiable framework, which are widely used constructs in physical simulations for common tasks such as handling collisions, evaluating boundary conditions, and building iterative solvers. Second, traditional physical simulation programs are written in imperative languages such as Fortran and C++.
- *Flexible indexing* – The programmer can manipulate array elements via arbitrary indexing, handy for the non element-wise operations such as numerical stencils and particle-grid interactions that are specific to physical simulations and not deep learning.

The initial work by Hu et al. [350] explains the design decisions and implementations nicely, along with illustrative examples in incompressible fluids, elastic objects, and rigid bodies with significant performance gains over comparable implementations in other DP (and deep learning) frameworks. Nonetheless these are on toy problems and much work remains to be done in complex real-world settings.

Julia δP: differentiable programming as a scientific ML *lingua franca* Dynamic programming languages such as Python, R, and Julia are common in both scientific computing and AI/ML for their implementation efficiency, enabling fast code development and experimentation. Yet there's a compromise reflected in most systems between convenience and performance: programmers express high-level logic in a dynamic language while the heavy lifting is done in C/C++ and Fortran. Julia [351], on the other hand, may give the performance of a statically compiled language while providing interactive, dynamic behavior. The key ingredients of performance are: rich type information, provided to the compiler naturally by multiple dispatch, code specialization against run-time types, and just-in-time (JIT) compilation using the LLVM compiler framework [369] – see Bezanson et al. [351] for details.

Potentially resolving the tradeoffs and annoyances of dynamic-static systems that couple Python and C, for example, Julia is a natural fit for differentiable programming towards scientific ML (not to mention probabilistic programming, geometric deep learning, and others we've touched on). The work by Innes et al. [352] is a DP system that is able to take gradients of Julia programs, making automatic differentiation a first class language feature. Thus Julia can support gradient-based learning and program transforms for the SI motifs and deep learning. We detailed several examples in the multi-physics and surrogate modeling motifs, in particular the Universal Differential Equations (UDE) system [68] (Fig. 8). Other illustrative examples include differentiable programming for finance, where contract valuation can be mixed with market simulations and other DP tools like neural networks; accelerated neural differential equations in financial time series modeling; and end-to-end automatic differentiation of hybrid classical-quantum systems (in simulation) [352].

Future directions

In the artificial intelligence field, differentiable programming has been likened to deep learning [370]. This does not do it justice, as DP can be viewed as a generalization of deep learning. The potential of DP is immense when taken beyond deep learning to the general case of complex programs – in existing programs taking advantage of the extensive amount of knowledge embedded within them, and enabling never before possible models and *in silico* experiments. DP

has the potential to be the lingua franca that can further unite the worlds of scientific computing and machine learning [68]. The former is largely governed by mechanistic equations and inefficient numerical solvers, and the latter with data-driven methods that learn high-dimensional spaces. DP provides exciting opportunity at the intersection, as we explored in the multi-physics motif.

DP tied to ML and simulation presents a relatively new area and there is much to do in both theory and practice. For the former, it would be valuable to derive techniques for working with non-differentiable functions like ReLU used in deep learning [371] or with non-smooth functions. It would also be useful to work with approximate reals, rather than reals, and to seek numerical accuracy theorems. For practical use, particularly in ML, it is important for DP to add explicit tensor (multi-dimensional array) types with accompanying shape analysis. In general, richer DP languages are needed to support wide ranges of types and computations – e.g. recursion, or programming with Riemannian manifolds (to accommodate natural gradient descent). There is also a need to move beyond Python as a host language, with computation runtimes magnitudes more efficient in Julia, C++ (e.g. DiffTaichi), and close-to-the-metal domain-specific languages (DSLs, see Fig. 1).

Just as AD is the practical power underlying DP, non-standard interpretation is the fundamental power underlying AD. Non-standard interpretation poses the question *how can I change the semantics of this program to solve another problem?*, the theory and formalism of which could provide rich developments beyond AD.

The Frontier

There are two nascent fields of AI and ML that we suggest have potentially massive upside for advancing simulation sciences and machine intelligence: open-endedness and machine programming. We now describe these as motifs in the SI stack.

8. OPEN-ENDED OPTIMIZATION

Open-endedness, or open-ended search, can be defined as any algorithm that can consistently produce novelty or increases in complexity in its outputs. Such open-endedness is a ubiquitous feature of biological, techno-social, cultural, and many other complex systems [372]. But its full algorithmic origins remain an open problem [373]. This open-endedness stands in contrast to most other kinds of optimization which generally seek to minimize loss functions, or otherwise reduce uncertainty [374, 375]. By instead seeking to increase complexity, open-ended search tends to increase uncertainty by definition [376]. We believe this is an extremely desirable property in optimization (and intelligence) as it has proven to be a source of robust behavior, and mechanistic innovation [377] as we will review below.

Open-ended learning algorithms are in general those with characteristics of life-long learning: adaptively and continually pose new objectives, adapt to environment changes, and avoid pitfalls such as local minima, deceptive loss functions [378], or catastrophic forgetting [379]. It follows that *open-ended optimization* is where components of a system continue to evolve new forms or designs continuously, rather than grinding to a halt when some sort of optimal, iteration threshold, or stable position is reached [380]. Open-ended optimization also has deep connections to fundamental issues in computer science. In theoretical analysis it has been shown to produce unbounded complexity, via computational universality [381]. Yet by these arguments we must embrace undecidability and irreducibility in the optimization [381], which has in turn lead to lead to practical results, such as the appearance of modularity in a memory system [382].

In general at least two major kinds of open-endedness can be distinguished: (1) processes that do not stop, and (2) processes that do not have specific objectives. It is straightforward to draw the connection to these features and to evolution. We believe open-ended algorithms, like those we will outline below, *must* serve as the mechanisms underlying evolution's creativity, at multiple levels of abstraction, from the evolution of protein sub-units, to protein networks, to the development eukaryotic systems, to speciation events, and perhaps the Cambrian explosion [382]. Rasmussen & Sibani [376] provide useful discussion to this end, by providing a means for disentangling the processes and algorithms of optimization and expansion within complex evolutionary processes. We aim to extend these concepts broadly in life sciences and other domains such as physics and engineering, as alluded to in the methods and examples discussed next.

Examples

Randomness and optimization Randomness is, in a sense, the simplest kind of open-ended search: A random process does indefinitely continue, and can generate different values which for a time leads to increases in entropy, and therefore output complexity. But of course for a finite tailed random process, entropy of the samples converges, and

so complexity converges. Nevertheless, the right amount of randomness (or noise) can bump an optimization process out of local minima, and so aid in improving the optimization [383]. Randomness offers a simple demonstration of how increasing complexity can later on aid an optimization whose objective is decreasing uncertainty. But, as the measured entropy of a random process does plateau, it is insufficient for the consistent introduction of complexity over the lifetime [384].

Some Bayesian optimization (BO) methods are examples of entropy-based search strategies, where acquisition functions intelligently balance explore-exploit search strategies [385] – recall an acquisition function guides the BO search by estimating the utility of evaluating the objective function at a given point or parameterization. A useful class of acquisition functions are information-based, such entropy search [386, 387] and predictive entropy search [388]. Yet these information-based methods aim to reduce uncertainty in the location of the optimizer, rather than increasing entropy continuously; BO is a key example of ML-based optimization that is orthogonal to open-endedness. Some BO strategies, and broadly Monte Carlo sampling, utilize search algorithms based on randomness. Rather, these are pseudo-random algorithms, such as Sobol sequences [389] and other quasi-Monte Carlo techniques [307, 390, 391].

Deceptive objectives and novelty search Deceptive optimization problems, are defined as problems for which the global optima appears highly suboptimal over the first few samples, and generally not solved by simple noise injection [392]. A concrete example of deception can be found by considering an agent in a maze. If the environment provides a proximity signal to the end of the maze, it would seem such a signal would make the maze trivial. But this is often not so. Consider that in most mazes an agent must walk away from the objective, for a time (to get around a bend, for example). In this case the proximity signal becomes a deceptive signal because while it is the optimal policy to follow the proximity signal overall, for short periods the winning agent must violate this optimal policy.

Stanley and colleagues [378] were the first to suggest that *open-ended novelty search*, defined as open-ended search to increase behavioral complexity, is sufficient to overcome deception. This is because the deceptive states are novel states. Other researchers have gone so far to suggest that novelty and curiosity (discussed below) are sufficiently good search/exploration algorithms that they can and should be used universally, in place of traditional objective-based optimization methods [393, 394, 395]. While novelty search does converge, there is early evidence novelty search interacts with traditional evolutionary algorithms to make evolvability–modularization and reusability–inevitable [396].

To increase searching efficiency in open-endedness, some efforts combine novelty search with *interactive evolution (IEC)*, where a human intervenes on the process to guide search [397, 398, 399]. IEC is particularly well suited for domains or tasks for which it is hard to explicitly define a metric, including when criteria are inherently subjective or ill-defined (e.g., salience, complexity, naturalism, beauty, and of course open-endedness).

Woolley & Stanley [398] implement novelty-assisted interactive evolutionary computation (NA-IEC) as a human-computer interaction system, where a human user is asked to select individuals from a population of candidate behaviors and then apply one of three evolutionary operations: a traditional IEC step (i.e. subjective direction by the human), a short term novelty search, or a fitness-based optimization. Experiments in a deceptive maze domain show synergistic effects of augmenting a human-directed search with novelty search: NA-IEC exploration found solutions in fewer evaluations, at lower complexities, and in significantly less time overall than the fully-automated processes.

Curiosity and self-training artificial systems Developmental artificial intelligence, defined as the invention and use of training algorithms which force agents to be self-teaching, relies on giving agents a sense of curiosity. Curiosity, loosely defined as learning for learning's sake [400], has been difficult to pin down experimentally both in psychological research [401, 402] and in AI research [403, 404, 405, 406, 407, 408]. That said, there is recent progress toward a unified mathematical view [393]. Details aside, the broad goal of *self-teaching-by-curiosity* is to lessen, or remove, the need for carefully curated datasets which are both expensive to create and often innately and unfairly biased [409, 410].

According to leading practitioners, effective self-teaching [411] requires an agent go about self-generating goals [407], discovering controllable features in the environment [412], and as a result make causal discoveries [413, 414]. It also requires agents to learn to avoid catastrophic mistakes [415], and handle environmental non-stationary [384]. Self-learning inverse models may also prove important [416]; in other words, self-teaching agents must generate self-organizing long-term developmental trajectories.

Quality-diversity and robustness to unknown in robotic systems The broad goal of *quality-diversity (QD)* algorithms is to produce a diverse set of high performing solutions to any given optimization problem [417]. Having diverse potential solutions for AI-driven research and decision making is desirable as it enables the practitioner or downstream algorithms to be robust in problem solving [417, 418]. In practice, this is done by combining novelty search (or other open-ended search approaches) and objective-based optimization. One of the most widely used algorithms is known as MAP-elites, developed by Mouret & Clune [419]. Clune and colleagues use MAP-elites to generate generalizable

action plans in robotic systems [420]. Remarkably though, QD also seems to solve a long-standing problem in robotics research: adaption to the unexpected [421]. Unlike agents in the natural world, robotic systems which work perfectly in trained environments, will fail frequently and catastrophically when even small but unexpected changes occur. They especially struggle when there are changes or damage to their appendages. However, robots trained using QD methods showed a remarkable ability to recover from the unknown, as well as to purposeful damage inflicted by the experimenters. In other words, seeking novelty, prioritizing curiosity, and building complexity, prepares agents to handle the unexpected [422]; open-ended learning and optimization can produce more robust AI agents.

Robotic systems for scientific discovery In addition to creating robust, flexible, and self-training robotics platforms, open-ended optimization methods have significant future potential for use in directly exploring scientific unknowns. For instance, Häse and colleagues have produced several objective-based optimization schemes for autonomous chemical experimentation, including approaches that combine objectives with user-defined preferences (with similar motivations as NA-IEC) [423], and in BO schemes mentioned above [424]. These are developed in the context of "Materials Acceleration Platforms" for autonomous experimentation in chemistry [425], where the specific implementations include he "Chemputer" [426] for organic synthesis (an automated retrosynthesis module is coupled with a synthesis robot and reaction optimization algorithm) and "Ada" [427] (Autonomous discovery accelerator) for thin-film materials. We continue discussing open-ended SI systems of this nature in the Integrations section later.

Lifelong machine learning and efficient memory A *lifelong learning* system is defined as an adaptive algorithm capable of learning from a continuous stream of information, with such information becoming progressively available over time and where the number of tasks to be learned are not predefined. In general the main challenge is *catastrophic forgetting*, where models tend to forget existing knowledge when learning from new observations [384]. This has been an active area of research in the neural network community (and connectionist models broadly, such as Hopfield Networks [428]) for decades, and most recently focused on deep learning. For a comprehensive review refer to Parisi et al. [429]. Here, in the context of open-endedness, we focus on the interplay of resource limitations and lifelong (continual) learning.

Memory-based methods write experience to memory to avoid forgetting [430], yet run into obvious issues with fixed storage capacity. Further, it is non-trivial to predefine appropriate storage without strong assumptions on the data and tasks, which by definition will evolve in continual learning settings. The open-endedness setting would exacerbate this issue, as increasing novelty over time would increasingly require writing new knowledge to memory. Central to an effective memory model is the efficient updating of memory. Methods with learned read and write operations, such as the differentiable neural computer (DNC) [339], use end-to-end gradient-based learning to simultaneously train separate neural networks to encode observations, read from the memory, and write to the memory. For continual learning problem domains, a DNC could hypothetically learn how to select, encode, and compress knowledge for efficient storage and retrieve it for maximum recall and forward transfer. Design principles for DNC and related external NN memories [431, 432, 433] are not yet fully understood; DNC is extremely difficult to train in stationary datasets, let alone in open-ended non-stationary environments[434].

Associative memory architectures can provide insight into how to design efficient memory structures, in particular using overlapping representations to be space efficient. For example, the Hopfield Network [428, 435] pioneered the idea of storing patterns in low-energy states in a dynamic system, and Kanerva's sparse distributed memory (SDR) model [436], which affords fast reads and writes and dissociates capacity from the dimensionality of input by introducing addressing into a distributed memory store whose size is independent of the dimension of the data. The influence of SDR on modern machine learning has led to the Kanerva Machine [437, 438] that replaces NN-memory slot updates with Bayesian inference, and is naturally compressive and generative. By implementing memory as a generative model, the Kanerva Machine can remove assumptions of uniform data distributions, and can retrieve unseen patterns from the memory through sampling, both of which behoove use in open-ended environments. SDR, initially a mathematical theory of human long-term memory, has additionally influenced architectures derived directly from neocortex studies, namely Hierarchical Temporal Memory (HTM) [439]. HTM is in a class of generative memory methods that is more like human and rodent intelligence than the others we've mentioned, and implements open-ended learning in sparse memory architecture. HTM networks learn time-based sequences in a continuous online fashion using realistic neuron models that incorporate non-linear active dendrites. HTM relies on SDR as an encoding scheme that allows simultaneous representation of distinct items with little interference, while still maintaining a large representational capacity, which has been documented in auditory, visual and somatosensory cortical areas [440]. Leveraging the mathematical properties of sparsity appears to be a promising direction for continual learning models, and open-ended machine intelligence overall.

Future directions

Curiosity, novelty, QD, and even memory augmentation algorithms are in practice quite limited in the kinds of complexity they generate, and all converge rather quickly such that complexity generation will plateau and then end. This is in striking contrast to even the simplest of biological systems (for example, E. Coli monocultures [376]). And the approaches for working, short term, and long term memories remain limited compared to the observed characteristics of their biological counterparts. We seem to be missing some central insight(s); this is a ripe area for extending the domain of artificial intelligence, where complexity sciences will have significant roles to play [441].

In the field of *artificial life (ALife)*, open-endedness is a crucial concept. ALife is the study of artificial systems which exhibit the general characteristics of natural living systems. Over decades of research practitioners have developed several kinds of Alife systems, through synthesis or simulation using computational (software), robotic (hardware), and/or physicochemical (wetware) means. A central unmet goal of ALife research has been to uncover the algorithm(s) needed to consistently generate complexity, and novelty, consistent with natural evolution [442]. Discovering and developing these algorithms would have significant implications in open-endedness and artificial intelligence broadly [443].

In the context of simulation more generally, a fully implemented open-ended system would mean not only solving optimization problems as posed by practitioners, but also posing new questions or approaches for practitioners to themselves consider. We believe this would be revolutionary. It would allow AI systems to leverage their prior expertise to create new problem formulations, or insights, and eventually, scientific theories. As important as investing in R&D to build such systems, we need to invest in understanding the ethical implications of this approach to science, and in monitoring the real-world, downstream effects.

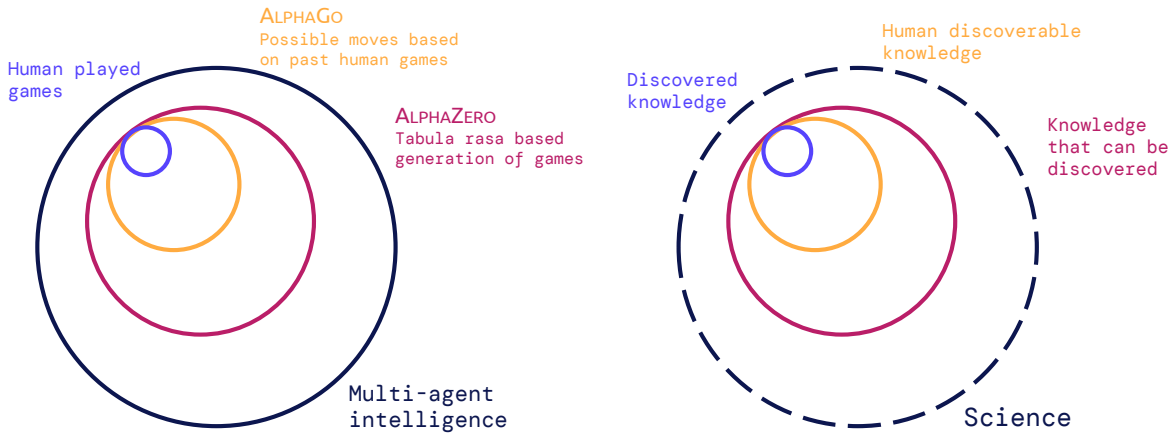

Figure 29: Left: Overall search space for multi-agent intelligence (black) – more precisely, perfect information games – and the subsets achieved by humans (purple), AlphaGo [245] (yellow), and AlphaZero [246] (red). Right: Open-ended search space for science (black), and subsets representing what has been achieved thus far (purple), what is achievable by humans (yellow), and what is achievable in general (red). (Diagrams are not to scale, inspired by [228].)

9. MACHINE PROGRAMMING

Machine programming (MP) is the automation of software (and potentially hardware) development. Machine programming is a general technology, in ways that are quite different than the other simulation intelligence motifs. As systems based on the simulation intelligence motifs grow in complexity, at some point these systems will necessitate automating aspects of software development. Machine programming can understand code across motifs, compose them into working systems, and then optimize these systems to execute on the underlying hardware – this interaction is illustrated in Fig. 1 at a high-level. Hence, while MP is not tied to any specific class of SI problems, MP will be integral as the glue that ties the other motifs together, serving as a bridge to the hardware.

The term *machine programming* made its debut in a paper by Gottschlich et al. [447] that defined the MP in terms of three distinct pillars: *(i) intention*, *(ii) invention*, and *(iii) adaptation* (Fig. 30). *Intention* is principally concerned with identifying novel ways and simplifying existing ways for users to express their ideas to a machine. It is also concerned

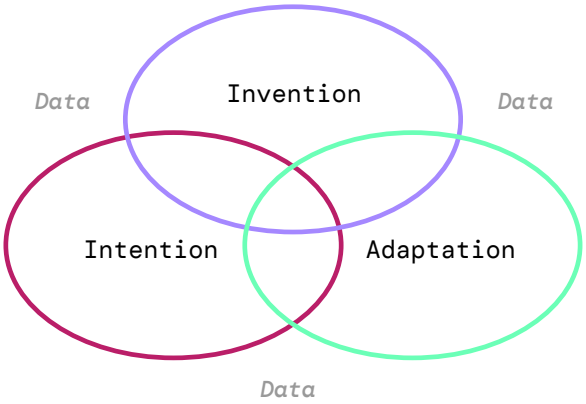

Figure 30: The "Three Pillars of Machine Programming": intention, invention, adaptation. Intention is quite literally to discover the intent of the programmer, and to lift meaning from software. Example sub-domains include program synthesis and inductive programming, which share territory with the invention pillar, for creating new algorithms and data structures. Sketch [444], FlashFill [445], and DeepCoder [446] are notable examples, with many others noted in text and in [447]. The adaptation pillar is for evolving in a changing hardware/software world, and more broadly in dynamic systems of data, sensors, and people [122] – examples include the DSL "GraphIt" [448] for computing on graphs of various sizes and structures on various underlying hardware, and "Tiramisu" [449] compiler for dense and sparse deep learning and data-parallel algorithms. (Figure reproduced from Gottschlich et al. [447])

with lifting meaning (i.e., semantics) from existing software and hardware systems [450, 451, 452]. *Invention* aims to construct the higher-order algorithms and data structures that are necessary to fulfill a user's intention. In many cases, inventive systems simply fuse existing components together to form a novel solution that fulfills the user's intent. However, there are some corner cases, which are likely to become more prevalent in the future, where an inventive system constructs a truly novel solution not yet been discovered by humans (e.g., Alam et al.'s inventive performance regression testing system [453] and Li and Malik's inventive optimization algorithm system [454]). *Adaptation* takes an *invented* high-order program and adapts it to a specific hardware and software system. This is done to ensure certain quality characteristics are maintained such as correctness, performance, security, maintainability, and so forth.

If trends continue, we anticipate programming languages will become more intentional by design. The byproduct of such intentionality is that such languages will be capable of more holistic automation of the inventive and adaptive pillars of MP, while simultaneously achieving super-human levels in key quality characteristics (e.g., software performance, security, etc.). Early evidence has emerged demonstrating this such as Halide [455], a programming language (more precisely a domain-specific language embedded in C++) designed for high-performance image and array processing code on heterogeneous modern hardware (e.g., various CPU and GPU architectures) and operating systems (Linux, Android, etc.). For example, Halide, in concert with verified lifting, has demonstrably improved the performance of Adobe Photoshop by a geometric mean of $3.36x$ for approximately 300 transpiled functions [450]. Moreover, other forms of intentional communication from humans to machines are already emerging, such as translation from natural language to programming language and translation of visual diagrams to software programs. Two embodiments of such systems are GitHub's Co-Pilot [456] and Ellis et al.'s automated synthesis of code for visual diagrams [457].

In machine programming, there appear to be at least two fundamental approaches for automation. The first is *statistical*, which consists of methods from machine learning and probabilistic analysis [453, 458]. The second is *rules-based*, which consist of approaches using formal methods that include techniques such as satisfiability solvers (SAT solvers) or satisfiability modulo theory (SMT) solvers [451, 459] – SAT is concerned with the problem of determining if there exists an interpretation that satisfies a given Boolean formula, and SMT is a generalization of SAT to more complex formulas involving real numbers, integers, and/or various data structures in order determine whether a mathematical formula is satisfiable.

As the field of MP matures, we are already seeing an emergence of systems that consist of both statistical and rules-based elements. A concrete example of this is Solar-Lezama et al.'s work fusing formal sketching techniques (where a *program sketch* is the schematic outline of a full program) with deep neural networks to improve computational tractability [444]. Another example is the Machine Inferred Code Similarity (MISIM) system by Ye et al., an automated engine for quantifying the semantic similarity between two pieces of code, which uses a deterministic and configurable

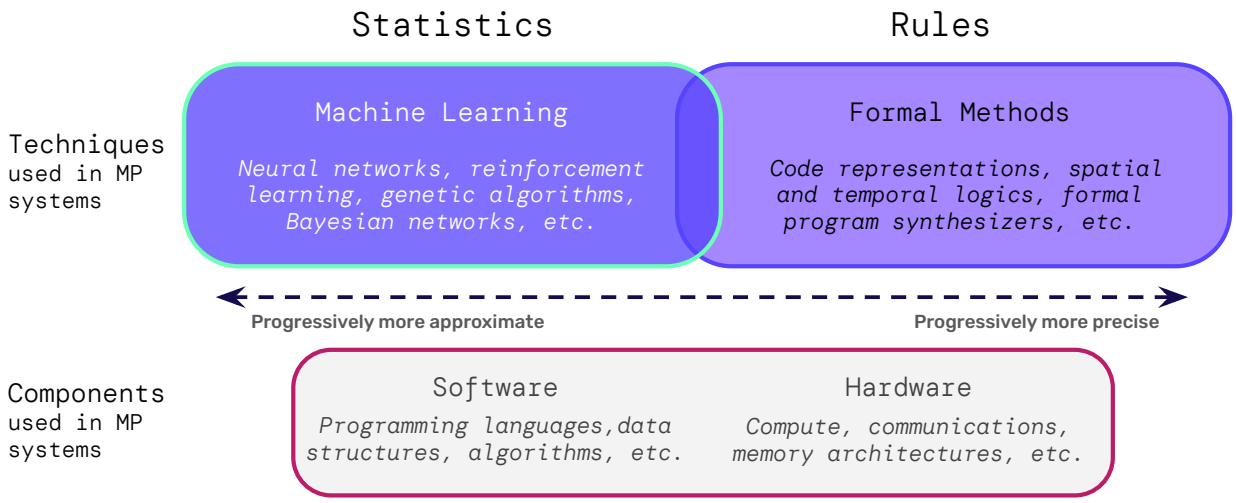

Figure 31: Illustrating the "bifurcated space" of machine programming, made up of *statistical* approaches, including various ML techniques like deep neural networks, and *rules-based* approaches, which include formal methods, deterministically reproducible representations, and so forth.

context-aware semantics structure (CASS) in conjunction with a learned similarity scoring system driven by a Siamese deep neural network with an extensible neural back-end [460].

The Department of Energy's 2020 report on Synthesis for Scientific Computing [461] provides an overview of where and how MP can have significant impact for scientific programming. In general, the consortium of experts anticipate increases in productivity of scientific programming by orders of magnitude by making all parts of the software life cycle more efficient, including reducing the time spent tracking down software quality defects, such as those concerned with correctness, performance, security, and portability [462, 463]. It is also believed that MP will enable scientists, engineers, technicians, and students to produce and maintain high quality software as part of their problem-solving process without requiring specialized software development skills, which today creates several choke points in organizations of researchers and engineers in many disconnected capacities.

Future directions

One of the core directions the field of MP is heading towards is *intentional programming*. Intentional programming is principally concerned with separating the intention of the program from the components that are used to realize that intention. A critical reason for this *separation of concerns* is that it tends to enable the machine to more exhaustively explore the inventive and adaptive components of software development – the programmer focuses on supplying the core ideas, while the machine handles the other two MP pillars, invention and adaptation.

A concrete embodiment of this is seen in the Halide programming language. Halide was designed with two primary components for two different types of programmers: *(i)* an algorithmic DSL for domain experts and *(ii)* the scheduling component for programmers with a deep understanding of building optimal software for a given hardware target. Due to the separation of the algorithm and schedule, the Halide team was able to automate the scheduling component to automatically extract performance. This is currently done using the Halide auto-scheduler using a learned cost model, which can synthesize more performant software (in the form of code, not neural networks) than the best humans it was pit against, which includes those individuals who developed the language itself [455].

We anticipate the field of MP to help accelerate the development and deployment of the SI stack. Referring back to the SI operating system conceptualization in Fig. 1, without MP one can imagine bottom-up constraints on the types of data structures and algorithms that can be implemented above the hardware layers This is not unlike a "hardware lottery", where a research idea or technology succeeds and trumps others because it is suited to the available software and hardware, not necessarily because it is superior to alternative directions [464]. Historically this has been decisive in the paths pursued by the largely disconnected hardware, systems, and algorithms communities; the past decade provides an industry-shaping example with the marriage of deep learning and GPUs. The various mechanisms of MP, however, provide opportunity to mitigate such hardware lottery effects for SI, and broadly in the AI and software fields. With MP in the SI operating system, as a motif that interfaces hardware and software, the integration of SI motifs can become

far more efficient (auto HW-SW optimizations and codesign), robust (less prone to human and stochastic errors), and scalable (flexibly deployable modules from HPC to the edge).

Simulation Intelligence Themes

We first discuss several common themes throughout the SI motifs, namely the utility of the motifs for solving inverse problems, the advantages (and scientific necessities) of implementing uncertainty-aware motif methods, synergies of human-machine teaming, and additional value to be realized from integrating motifs in new ways. Each of these themes may prove significant in the development of SI-based scientific methods. The discussion then continues to cover important elements for SI in practice, critically the data engineering and high-performance computing elements of SI at scale.

INVERSE-PROBLEM SOLVING

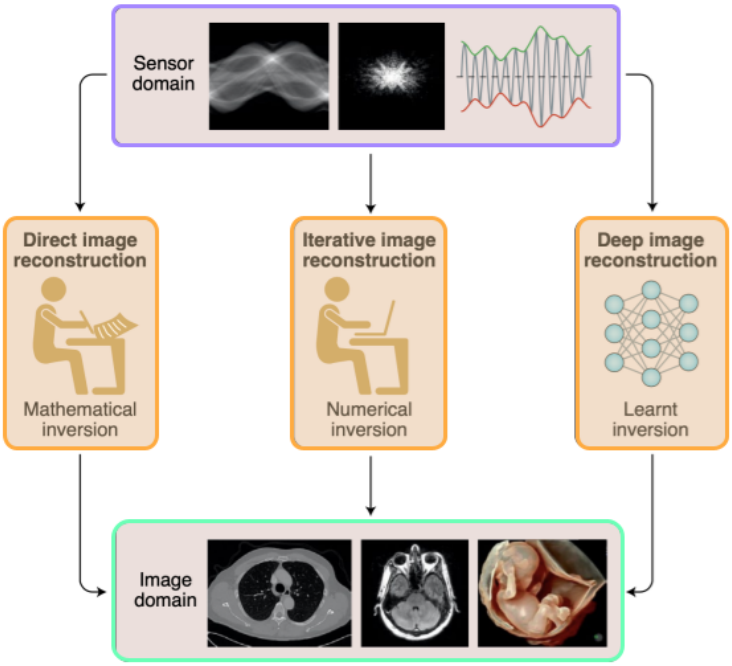

Figure 32: Three types of tomographic reconstruction methods as inverse problem-solving (orange). With direct or analytic reconstruction, a mathematical inverse of the forward transform is used to compute images from sensor data. In iterative reconstruction, the current image estimate is iteratively updated such that its forward transform gradually approaches the sensor data. In deep tomographic reconstruction, the inversion does not need to rely on any explicit transform model but is learnt from representative big data. (Original figure credit: Molly Freimuth (CT image); Christopher Hardy (mR image); Dawn Fessett (ultrasound image) [465]).

An *inverse problem* is one of inferring hidden states or parameters from observations or measurements. If the forward simulation (or data-generation) process is $x \rightarrow y$, we seek to infer x given y. In scientific settings, an inverse problem is the process of calculating from observations the causal factors that produced them. One typical example is computational image reconstruction, including MRI reconstruction, CT tomographic reconstruction, etc., as shown in Fig. 32. Most reconstruction methods could be classified as direct, analytical, or iterative reconstruction [465]. Based on the knowledge and information, a forward model can be derived to predict data given an underlying objective, either a simulation model or a mathematical/physical model. The reconstruction is addressed by computing the mathematical inverse of the forward model via nonlinear programming and optimization with regularization. Recently, deep learning based approaches provide a more promising way to inverse problem solving when conventional mathematical inversion

is challenging: Rather than relying entirely on an accurate physics or mathematical model, the new data-driven machine learning methods can leverage large datasets to learn the inverse mapping end-to-end.

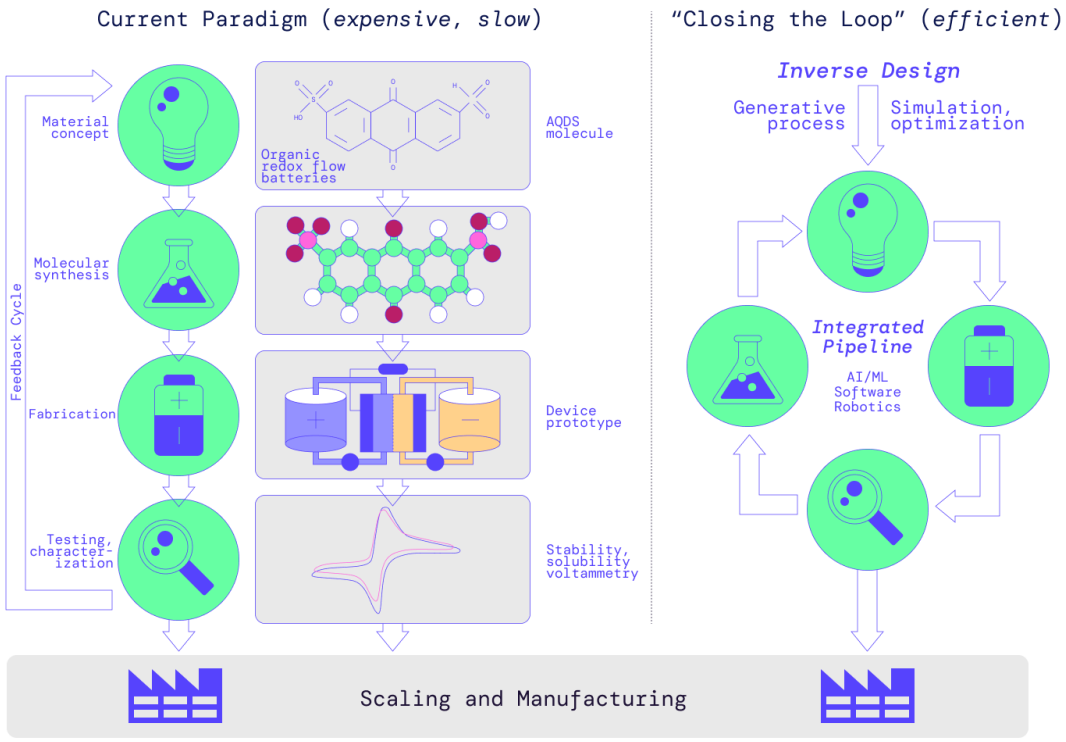

Figure 33: In many domains we can realize efficient inverse design brought about by simulation and AI technologies. This example schematic shows the current material discovery paradigm (here for organic redox flow batteries [466]) versus a closed-loop, inverse design counterpart *in silico* that maximizes efficiency and minimizes *in situ* resource-utilization.

Beyond the classic inverse problems, such as image reconstruction, inverse scattering, and computed tomography, much recent effort has been dedicated to the design of new materials and devices, and to the discovery of novel molecular, crystal, and protein structures via an inverse process. This can be described as a problem of *inverse design:* discovering the optimal structures or parameters of a design based on the desired functional characteristics – for example, computational inverse design approaches are finding many use-cases in optimizing nanophotonics [467, 468, 469] and crystal structures [470, 471, 472, 473]. Fig. 33 illustrates an example of how inverse design cycles can replace the existing, deductive paradigm that can be expensive and inefficient, and yield suboptimal solutions. In vast, complex spaces like materials science, inverse design can yield de novo molecules, foreshadowing what could be an SI-driven, inductive scientific method.

Inverse design differing from conventional inverse problems poses several unique challenges in scientific exploration. First, the forward operator is not explicitly known. In many cases, the forward operator is modeled by first-principles calculations, including molecular dynamics and density functional theory (DFT). This challenge makes inverse design intractable to leverage recent advances in solving inverse problems, such as MRI reconstruction [465], implanting on images through generative models with large datasets [474]. Second, the search space is often intractably large. For example, small drug-like molecules have been estimated to contain between 10^{23} to 10^{60} unique cases [475], while solid materials have an even larger space. This challenge results in obvious obstacles for using global search, such as Bayesian optimization [476], evolutionary strategies [477, 478, 479], or Bayesian inference via Markov Chain Monte Carlo (MCMC), since either method is prohibitively slow for high-dimensional inverse problems. Third, we encounter multi-modal solutions with one-to-many mapping. In other words, there are multiple solutions that match the desirable property. This issue leads to difficulties in pursuing all possible solutions through gradient-based optimization, which converges to a single deterministic solution and is easy trapped into local minima. Probabilistic inference also has limitations in approximating the complex posterior, which may cause the simple point estimates (e.g., maximum a

posterior (MAP)) to have misleading solutions [480]. Generally speaking, inverse design methods include Bayesian and variational approaches, deep generative models, invertible learning, and surrogate-based optimization.

Bayesian and variational approaches. From the inference perspective, solving inverse design problems can be achieved by estimating the full posterior distributions of the parameters conditioned on a target property. Bayesian methods, such as approximate Bayesian computing [481], are ideal choices to model the conditional posterior but this idea still encounters various computational challenges in high-dimensional cases. An alternative choice is variational approaches, e.g., conditional GANs [482] and conditional VAE [483], which enable the efficient approximation of the true posterior by learning the transformation between latent variables and parameter variables. However, the direct application of both conditional generative models for inverse design is challenging because a large number of data is typically required [484].

Deep generative models. Many recent efforts have been made on solving inverse problems via deep generative models [474, 480, 485, 486, 487, 488, 489]. For example, Asim et al. [474] focuses on producing a point estimate motivated by the MAP formulation and [485] aims at studying the full distributional recovery via variational inference. A follow-up study from [486] is to study image inverse problems with a normalizing flow prior which is achieved by proposing a formulation that views the solution as the maximum *a posteriori* estimate of the image conditioned on the measurements. For MRI or implanting on images, strong baseline methods exist that benefit from explicit forward operator [474, 480, 488].

Surrogate-based optimization. Another way is to build a neural network surrogate and then conduct surrogate-based optimization via gradient descent. This is more common in scientific and engineering applications [490, 491, 492, 493, 494], specifically for the inverse design purpose. The core challenge is that the forward model is often time-consuming so a faster surrogate enables an intractable search. A recent study in the scope of surrogate-based optimization is the neural-adjoint (NA) method [495] which directly searches the global space via gradient descent starting from random initialization, such that a large number of interactions are required to converge and its solutions are easily trapped in the local minima [496]. Although the neural-adjoint method boosts the performance by down selecting the top solutions from multiple starts, the computational cost is significantly higher, especially for high-dimensional problems [497].

Invertible models and learning. Flow-based models [366, 498, 499, 500, 501, 502], may offer a promising direction to infer the posterior by training on invertible architectures. Some recent studies have leveraged this unique property of invertible models to address several challenges in solving inverse problems [503, 504, 505]. However, these existing inverse model approaches suffer from limitations [495] in fully exploring the parameter space, leading to missed potential solutions, and fail to precisely localize the optimal solutions due to noisy solutions and inductive errors, specifically in materials design problems [506]. To address this challenge, Zhang et al. [507] propose a novel approach to accelerate the inverse design process by leveraging probabilistic inference from deep invertible models and deterministic optimization via gradient descent. Given a target property, the learned invertible model provides a posterior over the parameter space; they identify these posterior samples as an intelligent prior initialization which enables us to narrow down the search space. Then a gradient descent is performed to calibrate the inverse solutions within a local region. Meanwhile, a space-filling sampling [508] is imposed on the latent space to better explore and capture all possible solutions. More discussion on flow-based models follows in the context of "honorable mention motifs" later in this section.

Across sciences one can consider the simulation as implicitly defining the distribution $p(X, Z|y)$, where X refers to the observed data, Z are unobserved latent variables that take on random values inside the simulation, and y are parameters of the forward model. As we've suggested, there exist many use-cases and workflows for the inverse, where one wishes to infer Z or y from the observations $X = x$. The inverse design loop in Fig. 33 represents a class of this problem in a practical application. In many cases it is possible the solution to the inverse problem is ill-posed [509]: small changes in observed outputs lead to large changes in the estimate, implying the inverse problem solver will have high variance. For this reason, and to meet the needs of scientific applications in general, we often require uncertainty quantification, specifically given limited datasets [510, 511, 512] from high-fidelity simulations or experiments. Moreover, to fully assess the diversity of possible inverse solutions for a given measurement, an inverse solver should be able to estimate the complete posterior of the parameters (conditioned on an observation). This makes it possible to quantify uncertainty, reveal multi-modal distributions, and identify degenerate and unrecoverable parameters – all highly relevant for applications in science and engineering [513]. The challenge of inverse problems is also key in the context of causal discovery in science. Zenil et al. [514] describe formalisms connecting these challenges, based on the foundations of *algorithmic probability*, a framework where the parallel simulation of a large number of computable programs are used as generative models.

UNCERTAINTY REASONING

How to design a reliable system from unreliable components has been a guiding question in the fields of computing and intelligence [515]. In the case of simulators and AI systems, we aim to build reliable systems with myriad unreliable components: noisy and faulty sensors, human and AI error, unknown or incomplete material properties, boundary conditions, and so on. There is thus significant value to quantifying the myriad uncertainties, propagating them throughout a system, and arriving at a notion or measure of reliability.

Many of the recommended methods for each motif are in the class of probabilistic ML, first and foremost probabilistic programming, which naturally represents and manipulates uncertainty about models and predictions [27]. Probabilistic ML encompasses Bayesian methods that use probabilities to represent *aleatoric uncertainty*, measuring the noise inherent in the observations, and *epistemic uncertainty*, accounting for uncertainty in the model itself (i.e., capturing our ignorance about which model generated the data). For example, Zhu & Zabaras [516] develop a probabilistic encoder-decoder neural network in order to quantify the predictive uncertainty in fluid dynamics surrogate models, and recent extensions to physics-informed ML methods aim to implement Bayesian counterparts, such as Bayesian Neural ODE [517] and Bayesian PINN [44] – interestingly, some findings suggest that typical NN uncertainty estimation methods such as MC-dropout [518] do not work well with PDEs, which is counter to other PINN uncertainty quantification work [36], implying there is still much work to be done in this area. Gaussian processes, on the other hand, used in the same physics-modeling and emulation scenarios quantify aleatoric and epistemic uncertainties naturally [82].

Probabilistic numerics [43, 519] is a highly relevant class of methods that provide statistical treatment of the errors and/or approximations that are made en route to the output of deterministic numerical methods, such as the approximation of an integral by quadrature, or the discretized solution of an ordinary or partial differential equation [520]. The many *numerical algorithms* we've discussed in this paper estimate quantities not directly computable by using the results of more readily available computations, and the probabilistic numeric viewpoint provides a principled way to manage the parameters of such numerical computations. By comparison, epistemic uncertainty arises from the setup of the computation rather than the computation itself, and aleatory is concerned with the data. Hennig et al. [43] describe this class of methods in detail, along with practical uses such as Bayesian quadrature in astronomy. The more recent overview by Oates & Sullivan [520] provides more context, not to mention the suggestion that probabilistic programming will be critical for integration of probabilistic numerics theory and software, as well as enabling the tools from functional programming and category theory to be exploited in order to automatically compile codes built from probabilistic numerical methods (e.g., Ref. [521]).

In nearly all experimental sciences at all scales, from particle physics to cosmology, an important use of machine learning and simulation is to generate samples of labeled training data. For example, in particle physics, when the target y refers to a particle type, particular scattering process, or parameter appearing in the fundamental theory, it can often be specified directly in the simulation code so that the simulation directly samples $X \sim p(\cdot|y)$ [509]. In some cases this may not be feasible in the simulation code, but the simulation may provide samples $(X, Z) \sim p(\cdot)$, where Z are latent variables that describe what happened inside the simulation, but which are not observable in an actual experiment. Either case enables the researchers to generate accurate labeled training data. Yet while the data-generating simulator has high fidelity, the simulation itself has free parameters to be tuned and residual uncertainties in the simulation must be taken into account in downstream tasks. Comprehensively quantifying uncertainties in data-generators and simulators in general remains an area of active research – notably it must be interdisciplinary, as the uncertainty measures for ML do not necessarily suffice for physical sciences, and vice versa.

We have implied that development of robust AI and simulation systems require careful consideration of dynamic combinations of data, software, hardware, *and* people [122]. AI and simulation systems in sciences and decision-making often construct human-machine teams and other variations of cooperative AI [242], where it is valuable to calculate the various uncertainties in communications between combinations of humans, machines, agents, and sources of data and information broadly. Uncertainty-aware cooperative AI is a relatively new area of research, which we touch on more in the human-machine teaming section.

An additional direction for our work on uncertainty reasoning is to investigate how the motifs provide more powerful uncertainty methods than currently used in probabilistic ML, for instance the use of differentiable programming (more specifically autodiff) towards automatic uncertainty quantification. Models built in probabilistic programming frameworks are particularly useful because not only is uncertainty quantification trivial, the modeler is able to encode their expected uncertainties in the form of priors. Modern neural networks, on the other hand, are often miscalibrated (in the sense that their predictions are typically overconfident) due to ignoring epistemic uncertainty: NNs are typically underspecified by the data and thus can represent many different models that are consistent with the observations. Uncertainty estimation methods for NNs exist, although some research has demonstrated difficulties in translating these methods to physics-informed ML use-cases [44]. One promising direction is uncertainty-driven open-ended search. We've described the future benefits in optimization settings, and also suggest these same methods may be key in the

hypothesis-generation part of automating the scientific method – Kitano [228] describes these pursuits – and helping shape new scientific methods (for instance, built on principles of uncertainty and complexity, rather than falsifiability).

INTEGRATIONS

Throughout this paper we've suggested there's significant and valuable overlap between the module motifs. For instance, the use of multi-scale and agent-based modeling in applications such as systems biology and sociopolitics, the use of physics-infused ML for building surrogate models in multi-physics scenarios, using simulation-based causal discovery in agent-based network models, and more. We've also discussed some of the many ways that engine motifs integrate with and enable the module motifs. For instance, probabilistic programming languages (PPL) provide useful representations for multi-modal multi-scale simulation [288], agent-based modeling [522], semi-mechanistic modeling [307], and causal inference [523]. Another type of integration would be using probabilistic programs within the broader workflow of a module motif: e.g., within simulation-based inference (not to mention "human-machine inference" broadly) as shown in Fig. 10, or in scenarios for open-endedness. Same goes for the other engine motif, differentiable programming: Semi-mechanistic modeling, which is critical in multi-physics simulation amongst other scenarios, is only useful because of differentiable programming. And as we explained earlier, differentiable programming is a generalization of deep learning, and these neural network methods play significant roles in many motifs and workflows, e.g. multi-physics and inverse design, respectively. Not to mention, at the engine level, consider that a common approach for implementing PPL is embedding stochastic macros within existing languages such as Python or C++, and utilizing automatic differentiation features from differentiable programming. And there are valuable implications for leveraging both in the development of machine programming and open-endedness: for instance, auto-differentiating software to better enable learning and optimization in intentional programming, and utilizing PPL for readily encoding and tuning priors into probabilistic generative models in open-ended environments, respectively. Naturally, both machine programming and open-endedness will make heavy use of the module motifs such as causal discovery, ABM, multi-physics and multi-scale modeling.

The Nine Motifs of Simulation Intelligence (SI) are useful independently, yet when integrated they can provide synergies for advanced and accelerated science and intelligence. It is through these motif synergies that human-machine teams and AI agents can broaden the scientific method and potentially expand the space of achievable knowledge (Fig. 29). Implementation and some integration of the methods in existing machine learning frameworks and programming languages is doable, especially with growing ecosystems around frameworks based on autodiff and vector operations (notably PyTorch [348] and Tensorflow [347]), as well as probabilistic programming systems at multiple levels of abstraction (such as high-level Pyro [524] and PyMC3 [525] based on Python, and lower level Gen [526] based on Julia [351]). For future work, we aim to develop a more purpose-build framework for integrating the motifs. It would be advantageous to use a functional paradigm where models are represented (implicitly or explicitly) as functions; in homage to Pearl, functions could be pi functions (generative direction), lambda functions (inference direction), or coupled pairs of pi and lambda functions. Functional representations lend naturally to composition, abstraction, and multi-scale reasoning, and SI methods work by analyzing these functions. Such a framework would bring multiple motifs together by composing these analyses – e.g. probabilistic programming, multi-scale physics, differentiable programming, and causal discovery used together in a principled way in the same application. In general this can provide a unifying framework for machine intelligence, enabling the integration of our best methods for learning and reasoning.

Next we highlight several more motif integrations, amongst the many synergies through the SI stack to further develop:

Causal reasoning with probabilistic programs The execution of a simulator results in one or more simulation traces, which represent various paths of traversing the state space given the underlying data-generating model, which is not unlike the execution traces resulting from probabilistic programming inference. Formalizing the sequence of simulator states as a sequence of (probabilistic) steps, we can define a trace-based definition of causal dependency, which we introduce in an example below. With a simulator that produces a full posterior over the space of traces, as in probabilistic programming, we can attempt to draw causal conclusions with causal inference and discovery algorithms.

As alluded to in the probabilistic programming motif, the abstractions of probabilistic programming languages (PPL) provide ample opportunity to integrate the mathematics of causality with machine learning, which thus far has been non-trivial to say the least [198, 527]. There are several non mutually exclusive ways we suggest viewing this integration:

1. A probabilistic program is fundamentally a simulator that represents the data-generating process. As mentioned above, to address causal queries we must know something about the data generation process. A promising

approach is to build a probabilistic program in Box's model-inference-criticism-repeat loop [528] to arrive at a model that robustly represents the true data generating process, and use this data generator to infer causal relationships in the system – see Fig. 14. Said another way, one can conceptualize a structural causal model as a program for generating a distribution from independent noise variables through a sequence of formal instructions [201], which also describes a probabilistic program. The Omega PPL is a newer framework that provides initial steps towards this type of causal interpretation [529]. A resource-bounded approach to generate computable models was also introduced in the context of algorithmic information dynamics[514] leading to large sets of generative models sorted by algorithmic likeliness.

2. PPL by definition must provide the ability to draw values at random from distributions, and the ability to condition values of variables in a program via observations [286]. These two constructs are typically implemented within a functional or imperative host language as $sample$ and $observe$ macros, respectively. Similarly, one can readily implement the *do*-operator from the aforementioned do-calculus of causality [215, 216], to force a variable to take a certain value or distribution. This allows simulating from interventional distributions, provided the structural causal model (including the governing equations) is known. Witty et al. [530] have provided intriguing work in this direction, although limited both in expressiveness and scope – more development in robust PPL is needed. In general, via *do*-calculus or otherwise, the abstractions of PPL enable interventions, notably in the context of PPL-based simulators: the simulation state is fully observable, proceeds in discrete time steps [290], and can be manipulated such that arbitrary events can be prevented from occurring while the simulation is running.

3. PPL provide the mechanisms to easily distinguish intervention from observational inference, and thus implement counterfactual reasoning. A counterfactual query asks "what-if": *what would have been the outcome had I changed an input?* More formally: *what would have happened in the posterior representation of a world (given observations) if in that posterior world one or more things had been forced to change?* This is different from observational inference as it (1) fixes the context of the model to a "posterior world" using observational inference, but (2) then intervenes on one or more variables in that world by forcing each of them to take a value specified by the programmer or user. Interventions in the "posterior" world can cause variables — previously observed or otherwise — to take new values (i.e., "counter" to their observed value, or their distribution). Thus for counterfactual reasoning, we build a probabilistic program that combines both conditioning and causal interventions to represent counterfactual queries such as *given that X is true, what if Y were the case?* Recent extensions to the Omega language mentioned above aim to provide counterfactual queries via probabilistic programming, again with a version of Pearl's *do*-operator [531].

Causal surrogates and physics-infused learning Similar to the perspective that a probabilistic program encodes a causal data-generating process, one can view a system of differential equations as encoding causal mechanisms of physical systems. To be more precise, consider the system of differential equations $dx/dt = g(x)$ where x, $x(0) = x_0$, and $g(x) := nonlinearity$. The Picard–Lindelöf theorem [532] states there is a unique solution as long as g is Lipschitz, implying that the immediate future of x is implied by its past values. With the Euler method we can express this in terms of infinitesimal differentials: $x(t + \delta t) = x(t) + dtg(x)$. This represents a causal interpretation of the original differential equation, as one can determine which entries of the vector $x(t)$ cause the future of others $x(t + dt)$ [209]. A generalization of this approach in the context of algorithmic probability covers all computable functions including differential equations and mathematical models [514]. And in [527], it was shown that differentiability is not needed to perform an effective search optimization. The implication is then the algorithmic or physics-infused ML methods we discussed can represent the causal dynamics of physical systems, whereas neural networks fundamentally cannot, and differentiability is not the essential feature believed to allow neural networks to minimize loss functions – see Table 1, and also Mooij et al. [533] for a formal link between physical models and structural causal models. This provides theoretical grounding that algorithmic and physics-infused ML can be used as causal surrogate models which are a more accurate representation of the true causal (physical) processes than universal function approximators (including physics-informed neural networks). This insight also suggests that using algorithmic physics-infused surrogates may provide grounded counterfactual reasoning. We highlight several specific approaches:

Continuous-time neural networks are a class of deep learning models with their hidden states being represented by ordinary differential equations (ODEs) [534]. Compared to discretized deep models, continuous-time deep models can enable approximation of functions classes otherwise not possible to generate – notably the Neural ODE [78] which showed CTNN can perform adaptive computations through continuous vector fields realized by advanced ODE solvers. Vorbach et al. [535] investigate various CT network formulations and conditions in search of causal models. The popular Neural ODEs do not satisfy the requisite conditions for causal modeling simply because the can't account for interventions; the trained Neural ODE statistical model can predict in i.i.d. settings and learn from data, but it cannot predict under distribution shifts nor implicit/explicit interventions. Further, Neural ODEs cannot be used for counterfactual queries [533], which naturally follows from the causal hierarchy discussed previously (see Fig. 13).

	Predict in I.I.D. setting	Predict under shifts/interventions	Answer counterfactual queries	Obtain physical insight	Learn from data
Mechanistic / Physical	✓	✓	✓	✓	?
Structural causal (SCM)	✓	✓	✓	?	?
Causal graph	✓	✓	-	?	?
Physics-infused	✓	✓	?	✓	✓
Probabilistic graph (PGM)	✓	✓	-	?	✓
Physics-informed	✓	✓	-	-	✓
Statistical	-	-	-	-	✓

Table 1: Expanding on the taxonomy of models in Peters et al. [204] to show characteristics of the most detailed / structured models (top) to the most data-driven / *tabula rasa* models (bottom). The former, a mechanistic or physical model, is usually in terms of differential equations. The latter is often a neural network or other function approximator that provides little insight beyond modeling associations between epiphenomena. Causal models are in between with characteristics from both ends (as are the semi-mechanistic models we discussed in the surrogate modeling motif).

Liquid time-constant networks (LTCs) [536] do satisfy the properties of a causal model: The uniqueness of their solution and their ability to capture interventions make their forward- and backward- mode causal. Therefore, they can impose inductive biases on their architectures to learn robust and causal representations [537]. By this logic, we suggest UDEs can also be used as causal models of physical systems and processes. We also suggest the use of these "causal" models in practice may be more elusive than structural causal models that make assumptions and confounders explicit (whereas these NN-based approaches do not).

Physics-infused Gaussian processes in causal discovery – Earlier we described a "active causal discovery" approach where an agent iteratively selects experiments (or targeted interventions) that are maximally informative about the underlying causal structure of the system under study. The Bayesian approach to this sequential process can leverage a Gaussian process (GP) surrogate model, effectively implementing a special case of Bayesian optimization. The use of a physics-infused GP can provide robustness in the sense that the physical process being simulated is grounded as a causal surrogate – i.e., encoding causal mechanisms of said physical system – rather than a nonparametric function approximator. The several physics-infused GP methods we highlighted are *Numerical GPs* that have covariance functions resulting from temporal discretization of time-dependent partial differential equations (PDEs) which describe the physics [28, 29], modified Matérn GPs that can be defined to represent the solution to stochastic partial differential equations [30] and extend to Riemannian manifolds to fit more complex geometries [31], and the *physics-informed basis-function GP* that derives a GP kernel directly from the physical model [32].

Algorithmic Information Dynamics (AID) [538] is an algorithmic probabilistic framework for causal discovery and causal analysis. It enables a numerical solution to inverse problems based on (or motivated by) principles of algorithmic probability. AID studies dynamical systems in software space where all possible computable models can be found or approximated under the assumption that discrete longitudinal data such as particle orbits in state and phase space can approximate continuous systems by Turing-computable means. AID combines perturbation analysis and algorithmic information theory to guide a search for sets of models compatible with observations, and to precompute and exploit those models as testable generative mechanisms and causal first principles underlying data and processes.

Multi-fidelity simulation of electron paths for chemical reactions In an earlier example of differentiable and probabilistic programming we introduced how generative modeling in molecular synthesis can help mitigate inefficiencies in the standard design-make-test-analyze molecular discovery process. In that specific method we assumed a "reaction outcome predictor" exists [317]. That ability, to reliably predict the products of chemical reactions, is of central importance to the manufacture of medicines and materials, and to understand many processes in molecular biology. In theory all chemical reactions can be described by the stepwise rearrangement of electrons in molecules [539]. In practice this sequence of bond-making and breaking is known as the *reaction mechanism*, which can be described as several

levels of abstraction (corresponding do multiple spatial scales) [540]: computationally expensive quantum-mechanical simulations of the electronic structure at the lowest level, and rules-based methods for transforming reactant molecules to products in a single step that is often far too simplifying. A conceptual abstraction in the middle of these levels is used by chemists in practice: a qualitative model of quantum chemistry called *arrow pushing*, or sequences of arrows which indicate the path of electrons throughout molecular graphs [539]. Bradshaw et al. aim to leverage the domain knowledge embedded in arrow-pushing diagrams to machine-learn the mechanism predictions – in contrast to other ML approaches for directly predicting the products of chemical reactions [541, 542, 543, 544, 545]. Much like the interpretability the emerges from causal structural models versus purely data-driven ones, the resulting mechanism models are more interpretable than product prediction models.

The aim of their approach is to describe the problem of predicting outcomes of chemical reactions – i.e., identifying candidate products, given a particular set of reactants - as a generative process which simulates arrow-pushing diagrams conditional on the reactant set. The generative modeling approach comes in handy with the next "choice" of atom in the sequence, which is something domain-expert humans have essentially zero intuition about. Since this would simulate a mechanism, even though it is an approximate one, it is useful as a predictive model which does more than simply produce an answer (at the very least, the predicted mechanism can function as a type of explanation). The arrow pushing diagrams are sequences of arrows from atom to atom, which can be viewed as sequentially defining conditional distributions over the next atoms (i.e., a discrete distribution over atoms in the reactants), subject to some simple chemical constraints. The aim is to learn a distribution over model parameters for the probability of a series of electron actions that transform a set of reactant graphs into product graphs, which can also include graphs of reagents that aid the reaction process. In the current implementation the generative model is parameterized by a graph neural network and implemented in the autodiff (deep learning) library PyTorch. However, the well-practiced probabilistic programmer will recognize the direct analogy between series of conditional distributions and probabilistic programming inference, in particular sequential Monte Carlo based methods. The implementation in a PPL has notable advantages, for instance trace-based interpretability: different electron paths can form the same products (see Fig. 34), for instance due to symmetry, which can be inspected with PPL traces.

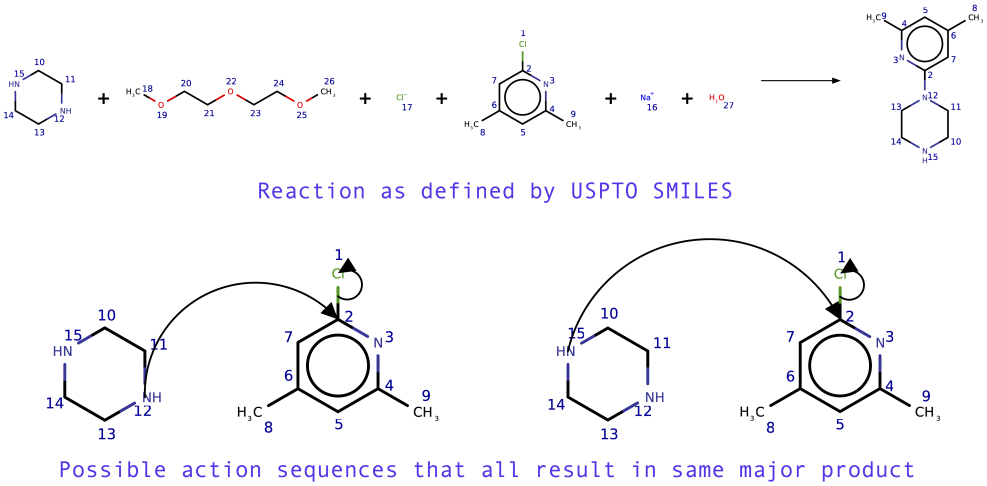

Figure 34: Example showing how different electron paths can form the same products, for instance due to symmetry. Thus modeling and interpreting the possible electron actions sequences are necessary for reliable evaluation of paths and prediction of products. The generative modeling approaches described in the text (using GNNs and probabilistic programming) are effective in overcoming this challenge. (Figure reproduced from Ref. [540])

With both GNN and probabilistic programming, we can more easily encode the constraints imposed by chemistry. For GNN, this is the geometric deep learning and equivarience properties we've discussed. Both also enable reasoning about multiple levels of fidelity: instead of simulating reactions at a low-level representation (involving atomic coordinates and other representations that are not understandable) we can simulate reactions at an intermediate granularity, one which we know domain experts are already familiar with and trust as an abstraction. Further for probabilistic programming, this again exemplifies the value of PP for "partially specified" models, which are the rich area between data-driven (black-box, deep learning) and fully-specified simulators. See the paper [540] for details, which has impressive results by training on large, unannotated reaction datasets (i.e., learning unsupervised), and further learns chemical properties it was not explicitly trained on.

SI-based materials acceleration platforms and open-ended search One property to exploit with open-ended search and optimization is the potential to discover entirely novel solutions, in spaces that otherwise would never have been explored. For instance, in socioeconomics as a variation on the multi-scale agent-based modeling AI Economist framework we discussed above – the dynamic optimization scheme suggests an alternative to typical objective-based convergence criteria (corresponding to Nash equilibria [546] inthe multi-agent setting) and can be extended with open-ended search algorithms to find non-obvious or even counter-intuitive socioeconomic solutions. A different sector has been gaining interest in AI and simulation tools to search for *de novo* solutions, pursuing "materials acceleration platforms" for autonomous experimentation in chemistry [425]. A fundamental goal of materials science and chemistry is to learn structure–property relationships and from them to discover *de novo* materials or molecules with desired functionalities. Yet the chemical space is estimated to be 10^{60} organic molecules [547] and the space of materials is even larger [425]. The traditional approach is based on human expert knowledge and intuition, resulting in a process that is slow, expensive, biased, and limited to incremental improvements. Even with high-throughput screening (HTS), the forward search process is limited to predesignated target areas [548]. Inverse design methods can provide significant efficiency gains [549], representing a key component for the molecule/hypothesis-generation phase of automating the discovery of chemicals and materials – see Fig. 33. As we described, inverse design focuses on efficient methods to explore the vast chemical space towards the target region, and fast and accurate methods to predict the properties of a candidate material along with chemical space exploration [472].

The focus in materials acceleration platforms has thus far been on objective-based optimization schemes for autonomous chemical experimentation, including approaches that combine objectives with user-defined preferences (with similar motivations as the NA-IEC approach we described for open-ended search) [423], and in Bayesian optimization schemes [424]. Specific implementations include he "Chemputer" [426] for organic synthesis (where an automated retrosynthesis module is coupled with a synthesis robot and reaction optimization algorithm) and "Ada" [427] (Autonomous discovery accelerator) for thin-film materials (with a purpose-built software package to coordinate researchers with lab equipment and robots [550]). These operate on closed-loop optimization schemes, which some suggest is a requirement for fully autonomous discovery systems [425, 550]. It is interesting to explore if open-ended search and optimization methods can advance these discovery platforms, and how they could be implemented in the environment. In such an expansive space of molecules, one can leverage open-ended search to not only explore more broadly, but also in ways that increase uncertainty (and entropy) towards potentially unknown territories that can match target characteristics. We are not yet aware of work in this direction, and even so it may call for a rethinking of the broader discovery pipeline. That is, the closed loop setting of current approaches may not support open-ended search in optimal ways. Continual learning methods may be usable, as the modeling and search algorithms aim to improve on subsequent passes through the cycle, but that is still very much objective-based and does not allow for exploration with ongoing, targeted increases in entropy. Perhaps an offline process for open-ended search that is decoupled from the main process is needed, along with some mechanism for information flow between the two (so the open-ended search can learn from the materials synthesis process, for example). And to consider integrating robotics with open-endedness would introduce additional challenges (not to mention risks). One can potentially look to surrogate modeling to alleviate some of the challenges implementing open-ended search and optimization, but at this point we are well into speculative territory and there is still much platform engineering and baseline experiments to be done.

It's straightforward to connect materials acceleration platforms with nearly all the SI motifs, as well as inverse design and human-machine teaming workflows. For instance, semi-mechanistic modeling can be advantageous for incorporating domain expert priors into the AI-driven search, and well as physics-infused learning. A significant challenge with chemical and materials design is the translation of properties towards production scaling, which can potentially be alleviated with improved multi-scale modeling techniques. With such a massive, complex environment that cannot possibly be understood by humans, causal modeling and inference can prove valuable for interpreting the solution space and navigating cause-effect relationships in the overall process.

Inducing physics programs with open-ended learning One would suggest that open-ended methods train in real-time rather than batches or episodes like mainstream deep learning – online learning could be a requirement of any model developed for open-ended use. However, "DreamCoder" [334] provides an interesting counterexample with "wake-sleep" [551] probabilistic program learning.

DreamCoder is a system that learns to solve problems by writing programs – i.e., a program induction perspective on learning, where learning a new task amounts to searching for a program that solves it. This perspective provides a number of advantages: Nonetheless practical use is limited, as applications of program induction require not only carefully engineered domain-specific language (DSL) with strong biases/priors on the types of programs to find, but also a search algorithm hand-designed to exploit that DSL for fast inference as the search space is otherwise intractable. The wake-sleep approach in DreamCoder resolves these issues by learning to learn programs: its learned language defines a generative model over programs and tasks, where each program solves a particular hypothetical task, and its

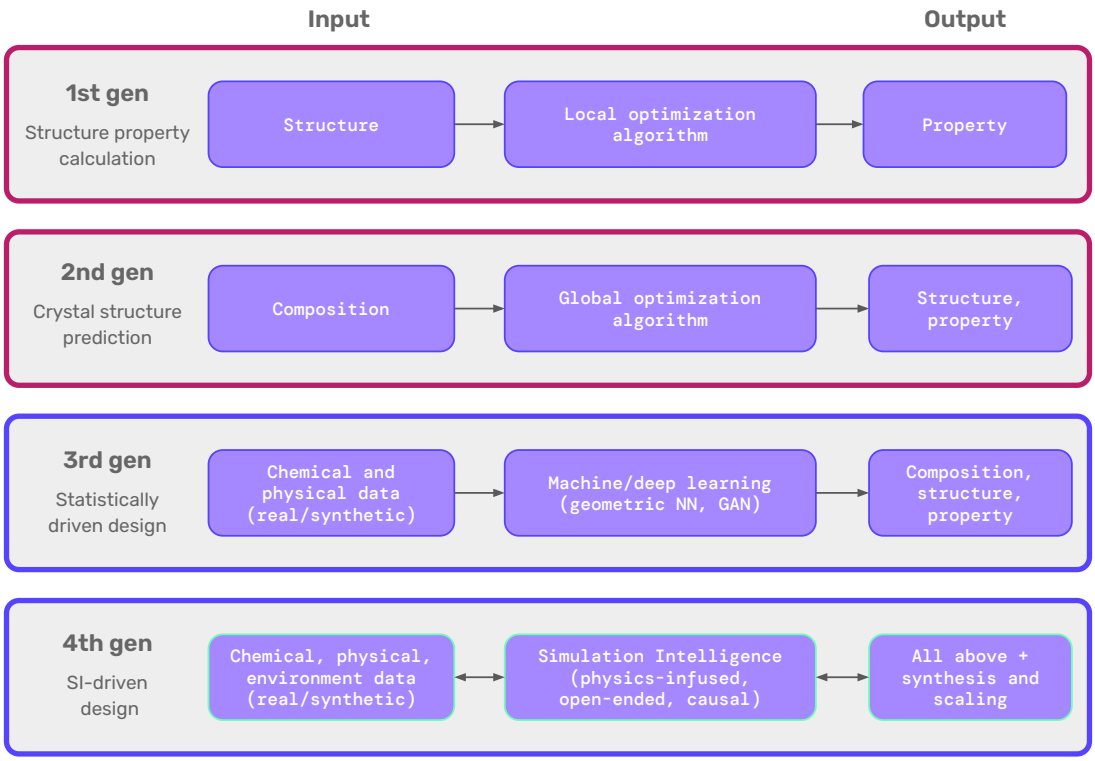

Figure 35: Evolution of the research workflow in computational chemistry (extended from the "3rd generation" graphic in [51]). The standard paradigm in the first-generation approach is to calculate the physical properties of an input structure, which is often performed via an approximation to the Schrödinger equation combined with local optimization of the atomic forces. In the second-generation approach, by using global optimization (for example, an evolutionary algorithm), an input of chemical composition is mapped to an output that contains predictions of the structure or ensemble of structures that the combination of elements are likely to adopt. The third-generation approach is to use machine-learning techniques with the ability to predict composition, structure, and properties provided that sufficient data are available and an appropriate model is trained. The outdated first and second generations (red outline) are replaced by the third and fourth generation approaches (blue outline) that excel in efficiency and producing optimal solutions. We suggest these two vectors – *efficiency and optimality* – are where SI-based methods can bare fruit. For efficiency, in practice we see a significant roadblock in the synthesis and scaling steps of new chemicals and materials – see Fig. 37 – which can be addressed with the integrated, closed loop approaches of SI, such as simulation-based inference (and "human-machine inference" in Fig. 12), physics-*infused* learning, causal discovery (Fig. 14), and inverse problem solving (Fig. 33). This interconnectedness is loosely reflected in the figure with forward-backward arrows to contrast with the standard forward computational approaches. As for optimality, an overarching goal for SI is to both expand and intelligently navigate the space of potential solutions. The implementation of open-ended optimization in this context has exciting prospects. It remains to be seen where and how "quantum learning" (i.e., quantum computing, ML, and data) plays a role in this evolution. We discuss quantum learning in the Accelerated Computation and Honorable Mention sections.

inference algorithm (parameterized by a neural network) learns to recognize patterns across tasks in order to best predict program components likely to solve any given new task. This approach produces two distinct kinds of domain expertise: (1) explicit declarative knowledge, in the form of a learned DSL that captures conceptual abstractions common across tasks in a domain; and (2) implicit procedural knowledge, in the form of a neural network that guides how to use the learned language to solve new tasks, embodied by a learned domain-specific search strategy [334]. This succinct description does not do DreamCoder justice, as the paper thoroughly discusses the approach and learned representations, and presents illustrative examples like that in Fig. 36. The authors show this setup allows learning to scale to new domains, and to scale within a domain provided there are sufficiently varied training tasks. DreamCoder is open-ended in the sense that it learns to generate domain knowledge ceaselessly, implements a continual learning algorithm, and doesn't have a specific optimization objective. Another important distinction from deep neural networks is the system

is not learning *tabula rasa*, but rather starting from minimal bases and discovering the vocabularies of functional programming, vector algebra, and physics.

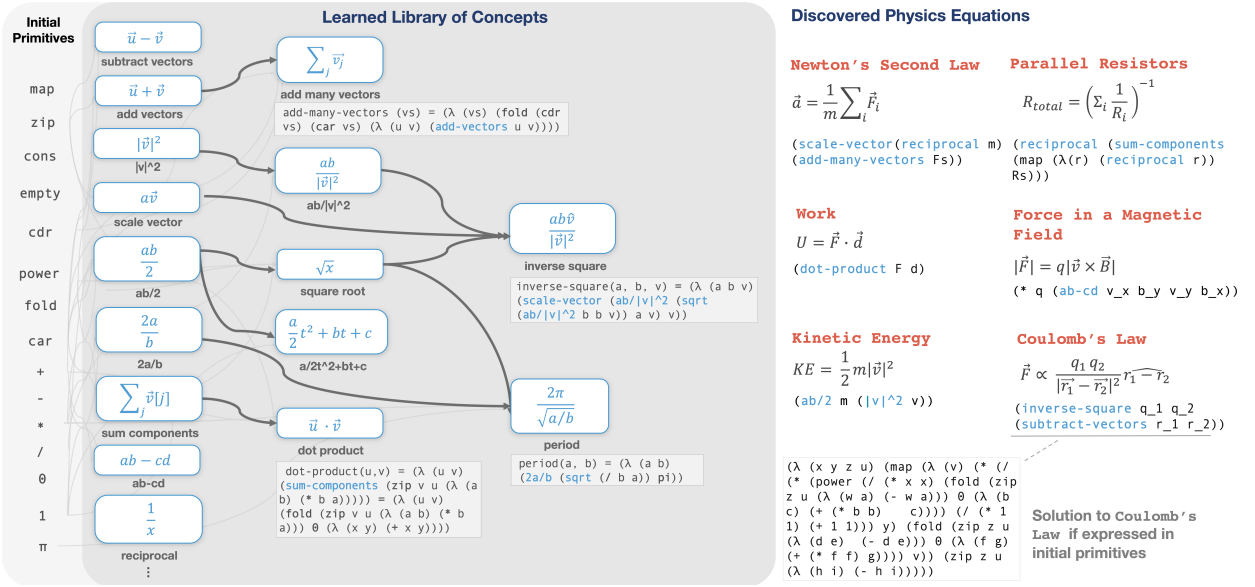

Figure 36: Learning a language for physical laws starting with recursive list routines such as map and fold. DreamCoder observes numerical data from 60 physical laws and relations, and learns concepts from vector algebra (e.g., dot products) and classical physics (e.g., inverse-square laws). (Original figure from Ellis et al. [334])

Physical computing Here we highlight several promising directions for improving physical computation with the incorporation of SI methods: optimizing the design of sensors and chips.

ML-enabled intelligent sensor design is the use of inverse design and related machine learning techniques to optimally design data acquisition hardware with respect to a user-defined cost function or design constraint. Here is a general outline of the process (motivated by Ballard et al. [552]):

1. Standard engineering practices produce an initial design α_0, or a randomized initialization of parameters and spatial configuration (in terms of transduction elements or sensing components, for example).
2. Acquisition of raw sensing data X_i, y_i from the prior α_{i-1}, and data preprocessing and transformations to X_i' such that the data is amenable to machine learning (e.g., normalized, Gaussian noise, etc.).
3. Train a ML model (typically a neural network or Gaussian process, potentially a probabilistic program) to output sensing results as $\hat{y}_i$
4. A cost function $J(y_i, \hat{y}_i)$ is used to evaluate the learned model, using the ground-truth sensing information y_i – here it is useful to develop a problem-specific cost function with physics-informed constraints (see the multi-physics motif section).
5. The results from step (4) inform a redesign α_{i+1} that consists of adjusting or eliminating the less informative/useful features, and possibly replacing such features with alternative sensing elements. Ideally one can use a Bayesian optimization (BO) outer-loop to inform the subsequent design, where the learned NN or GP serves as the surrogate (as in the surrogate modeling motif). Future work may investigate BO-based neural architecture search methods (e.g. [114]) applied to sensor components and modules.
6. Repeat from step (2) for $i = 1, .., N$, where termination conditions may be the number of budgeted cycles (N), convergence on design or outcome, or achieving a down-stream cost function. Depending on the surrogate model choice and optimization scheme, it may be useful to employ transfer learning techniques to reduce the data burden of subsequent iterations.

Such an approach can drastically improve the overall performance of a sensor and broader data ingestion/modeling system, perhaps with non-intuitive design choices. Consider an intriguing example in the field of synthetic biology, aiming to intelligently design RNA sequences that synthesize and execute a specific molecular sensing task: Angenent et al. [553] developed a surrogate-based design framework with a neural network trained on RNA sequence-to-function

mappings, resulting in an *in silico* design framework for engineering RNA molecules as programmable response elements to target proteins and small molecules. The data-driven approach outperforms the prediction accuracy resulting from standard thermodynamic and kinetic modeling. In general data-driven, ML-based sensor designs can outperform "intuitive" designs that are solely based on analytical/theoretical modeling, but there are non-trivial limitations: generally large, well-characterized training datasets are needed to engineer and select sensing features that can statistically separate out various inherent noise terms or artifacts from the target signals of interest. Not to mention the the curse of dimensionality, where the high-dimensional space of training data may drown-out the meaningful correlations to the target sensing information. See Ballard et al. [552] for more discussion on ML-enabled intelligent sensor design.

ML-based chip design aims to automate and optimize the various processes in designing, fabricating, and validating computer chips. In particular, recent work has made significant strides in ML-based "chip floorplanning" [554], or designing the physical layout of a computer chip. This slow, costly process involves placing hypergraphs of circuit components (such as macros (memory components) and standard cells (logic gates such as NAND, NOR and XOR)) onto chip canvases (two-dimensional grids) so that performance metrics (e.g., power consumption, timing, area, and wire length) are optimized, while adhering to hard constraints on density and routing congestion. Engineers approach this manual design task over the course of months per chip layout, where performance evaluation of each design takes multiple days. Mirhoseini et al. [554] developed an reinforcement learning (RL) approach, not unlike those described in the agent-based modeling motif, by framing the task as a sequential Markov decision process (MDP) as follows:

- States: hypergraph (as an adjacency matrix), node features (width, height, type), edge features (number of connections), current node (macro) to be placed, and metadata of the netlist graph (routing allocations, total number of wires, macros and standard cell clusters).
- Actions: all possible locations (grid cells of the chip canvas) onto which the current macro can be placed subject to predefined hard constraints on density or blockages.
- Transitions: probability distribution over next states, given a state and an action.
- Rewards: always zero but for the last action, where the reward is a negative weighted sum of proxy wirelength, congestion, and density. (See the paper for details [554])

A graph convolutional NN is used to learn feature embeddings of the macros and other hypergraph components – this architecture provides advantageous geometric properties for this design space, which we discuss more in the "honorable mention motifs" later. The benchmark results of this approach show the generation of chip floorplans that are comparable or superior to human experts in under six hours, compared to multiple months of manual human effort. This is a nice practical example of AI-driven knowledge expansion in Fig. 29: the artificial agents approach the problem with chip placement experience that is magnitudes greater in size and diversity than any human expert.

As alluded to above, this is one step of many in a resource-intensive process of computer chip development. There are many ways automation and human-machine teaming can potentially optimize and accelerate the process, including additional ways simulation-AI can advance chip design: for example, the SimNet [19] platform of NN-solvers has been applied to the problem of design optimization of an FPGA heat sink, a multi-physics problem that can be approached with multiple parallel ML surrogates (with one NN for the flow field and another NN for the temperature field).

Physical Neural Networks – In the simulation intelligence spirit of minimizing the gaps between the *in silico* and *in situ* worlds, *physical neural networks (PNNs)* are an intriguing direction for physical learning hardware. That is, PNNs describe hierarchical physical computations that execute deep neural networks, and trained using backpropagation through physics components. The approach is enabled by a hybrid physical-digital algorithm called *physics-aware training (PAT)*, which enables the efficient and accurate execution of the backpropagation algorithm on any sequence of physical input-output transformations, directly *in situ* [555]. For a given physical system, PAT is designed to integrate simulation-based and physical modeling in order to overcome the shortcomings in both. That is, digital models (or simulations) describing physics are always imperfect, and the physical systems may have noise and other imperfections. Rather than performing the training solely with the digital model (i.e., gradient-based learning with the simulator), the physical systems are directly used to compute forward passes, which keep the training on track – one can view this as another type of constraint in the physics-infused learning class of methods we discussed earlier. Naturally this necessitates a differentiable simulator of the physical system, one to be built with differentiable programming and perhaps additional SI motifs. Wright et al. [555] trained PNNs based on three distinct physical systems (mechanical, electronic, and optical) to classify images of handwritten digits, amongst other experiments, effectively demonstrating PAT and the overall physical computing (or physical learning) concept. They also describe in depth the mathematical foundations of all components of the approach, including the implementation of autodiff for physics-aware training. The overall approach is very nascent but could expand possibilities for inverse design (such as inverse design of dynamical engineering systems via backpropagation through physical components and mechanisms. Physical neural networks may also facilitate cyber-physical simulation environments with a variety of machines and sensor systems, and may

lead to unconventional machine learning hardware that is orders of magnitude faster and more energy efficient than conventional electronic processors.

HUMAN-MACHINE TEAMING

The stack in Fig. 1 indicates the motifs and applications interact with people of diverse specialties and use-cases. Not to mention the human-machine inference methods we've described (in simulation-based inference, causal reasoning, and other motifs) call for human-in-the-loop processes. To this end we define *human-machine teaming (HMT)* to be the mutually beneficial coordination of humans and machine intelligence systems (which include AI and ML algorithms and models, various data and information flows, hardware including compute architecture and sensor arrays, etc.). HMT overlaps with the more well-established field of *human-computer interaction (HCI)*, which integrates the disciplines of computer science, cognitive science, and human-factors engineering in order to design technologies that enable humans to interact with computers in efficient and novel ways.[iv]

Common practice in machine learning is to develop and optimize the performance of models in isolation rather than seeking richer optimizations that consider human-machine teamwork, let alone considering additional human-in-the-loop challenges affecting model performance, such as changes in policies or data distributions. Existing work in HMT includes systems that determine when to consult humans, namely when there's low confidence in predictions [556, 557, 558], or informing the human annotator which data to label for active learning [559, 560].

HMT research addresses several fundamental issues with applicability across domains of science and intelligence:

- How humans and machines communicate
- How to model the relationship between humans and machines for accomplishing closed feedback loops
- How to quantify and optimize human-machine teamwork
- Machines learning how to best compliment humans [561]
- How human-machine dynamics can influence and evolve the characteristics of errors made by humans and AI models, especially in continual learning contexts
- How to quantify human-machine uncertainties and predict reliability, especially in the context of broader and dynamic systems [122]
- How to provide theoretical guarantees for various HMT subclasses (e.g., online optimization of experiments, active causal discovery, human-machine inference (Fig. 12)), and how to empirically validate them
- How to capture and express the human-machine interactions for a particular application domain – *where do general methods suffice, and where are domain-specific algorithms and interfaces needed?*
- Synergistic human-AI systems can be more effective than either alone – we should understand this tradeoff precisely, which will likely vary in metrics and dimensions depending on the domain and use-case
- Laboratory processes and equipment are currently designed for the traditional, deductive scientific method, where tools assist human scientists and technicians. Simulation intelligence provides the opportunity to rethink this human-machine relationship towards new science methods that are inductive, data-driven, and managed by AI (beyond standard automation).

The standard HMT approach is to first train a model m on data x for predicting y, then m is held fixed when developing the human-interaction components (and broader computational system) and when training a policy q for querying the human. One of the more glaring issues with this general approach is the introduction of q can cause the x and y distributions to shift, resulting in altered behavior from m that necessitates retraining. Joint training approaches could mitigate this concern, and further provide a more optimal teaming workflow. For instance, Wilder et al. [561] propose several joint training approaches that consider explicitly the relative strengths of the human and machine. A *joint discriminative* approach trains m while learning q end-to-end from features to decisions, and a a new *decision-theoretic* approach to additionally construct probabilistic models for both the AI task and the human response. The latter allows a follow-up step that calculates the expected value of information for querying the human, and joint training optimizes the utility of the end-to-end system.

[iv] We intentionally use the name "machine" in a general sense, overlapping with and extending beyond HCI, human-AI interaction, and human-robot interaction. The development of HMT is to include the variety of machines we've discussed in this paper, from *in silico* agents and algorithms, to *in situ* nuclear fusion reactors and particle accelerators, and of course the nascent field of machine programming.

In addition to quantifying the value of information in human-machine dynamics, using probabilistic ML methods can also enable the quantification and propagation of uncertainties in HMT (and overall throughout SI-based workflows for automating science – Fig. 37). We've described candidate methods in the previous section, and here point out another class of methods for uncertainty estimation in the human-machine coalition setting: *Evidential learning* maps model inputs to the parameters of a Dirichlet distribution over outputs (classes), where smaller parameter values represent less evidence for a class and thus produce a broader distribution representing greater epistemic uncertainty [562, 563]. Evidential learning enjoys a direct mapping to *subjective logic*, a calculus for probabilities expressed with degrees of uncertainty where sources of evidence can have varying degrees of trustworthiness [564]. In general, subjective logic is suitable for modeling and analysing situations where the available information is relatively incomplete and unreliable, which can be utilized in HMT scenarios for uncertainty-aware logical reasoning. One can extend this to trust calibration, a valuable yet overlooked factor in many HMT scenarios: *automation bias* and *automation complacency* can occur when humans too readily accept AI or machine outputs. In these cases, humans either replace their own thinking or judgement, or are less sensitive to the possibility of system malfunctions [565, 566]. Conversely, *algorithm aversion* occurs when humans disregard algorithms for lack of trust, despite the algorithms consistently outperforming humans. In general for feedback systems powered by ML, one cannot simply improve the prediction components and reliably improve performance. As in control theory, the "Doyle effect" [567, 568] suggests that an improved prediction system can increase sensitivity to a modeling or data flaw in another part of the engineering pipeline – this certainly holds for human-in-the-loop ML, and scientific ML pipelines in general.

The HMT concerns are relevant and perhaps exacerbated in the broader perspective of a full workflow like that of Fig. 37, where we illustrate a scientist in the context of a largely autonomous workflow for AI-driven science.

Simulation Intelligence in Practice

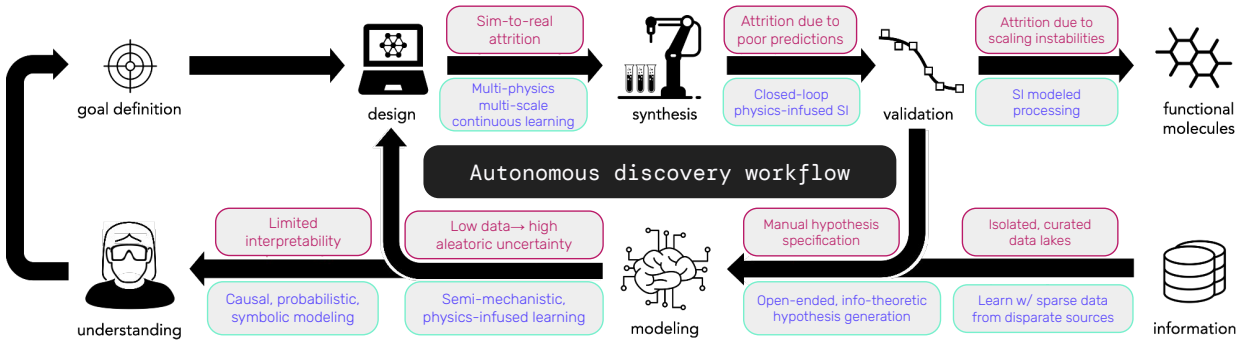

Figure 37: The AI field largely focuses on modeling and design, which represent only a subset of the AI and data systems required for highly-autonomous science. Here we show one high-level example for molecule discovery, where modeling and design must be integrated with various information flows and coordinate with upstream and downstream tasks such as human-input and material synthesis. Plugging in mainstream AI technologies would not suffice, with shortcomings noted in red along the workflow phases. Also noted in purple (and green) are corresponding advantages that SI motifs and methods alternatively bring to the workflow. The additional use of the uncertainty-aware methods we discussed would improve the reliability of such a system. From this view one can also envision some of the SI data challenges we describe. For example, the multitude of data consumers and generators in the workflow amplifies the risks of data inconsistencies and errors, which can be mitigated with proper data versioning and provenance. Further, provenance and data readiness steps are necessary to ensure reproducibility of the experiment process and results. We stress this is a simplified workflow that assumes data fusion and synthetic data needs are taken care of, and doesn't shed light on the significant data engineering throughout each phase.

The implementation and integration of simulation intelligence in workflows such as this can address the noted issues in important ways. For instance, the semi-mechanistic modeling and physics-infused learning we shared earlier are essential in the modeling-understanding phase, and are also critical in the sim-to-real transfer phase(s); we suggest these need to be integrated in closed-loop ways that coordinate learning and info-sharing between upstream and downstream phases. The inverse design approaches we presented (such as Fig. 33) could be useful to this end, and inverse problem solving may prove useful for mitigating losses in the synthesis steps – this is an area of future SI work for us. More

generally it can be advantageous to reshape such workflows from the perspective of SI. For instance, human-machine inference can in essence replace the left side of the diagram, and SI-based inverse design can flip/rewire much of the inner modeling-design-synthesis-validation loop. Nonetheless the data and engineering challenges remain non-trivial in practice, which we discuss next.

Data-intensive science and computing

As we've described, scientific datasets are different from the data typically available to ML research communities; scientific data can be high-dimensional, multi-modal, complex, sparse, noisy and entangled in ways that differ from image, video, text and speech modalities. Scientific questions often require the combination of data from multiple sources, including data to be fused from disparate and unknown sources, and across supercomputing facilities. Further, many science applications where ML and AI can be impactful are characterized by extreme data rates and volumes, highly diverse and heterogeneous scenarios, extremely low latency requirements, and often environment-specific security and privacy constraints.

For these challenges and more, data processing, engineering, sharing, governance, and so on are essential to scientific AI and simulation intelligence, which are inherently *data-intensive*: problems characterized by data volume, velocity, variety, variability (changes both in the structure of data and their underlying semantics, especially in real-time cases), and veracity (data sources are uncontrolled and not always trustable) [569]. Admittedly we do not discuss nearly enough the importance of data in the SI motifs and in simulation sciences broadly – one or more survey papers on this is warranted. Here we highlight several key concepts and references.

Data challenges and opportunities in merging scientific computation and AI/ML In recent years there have been numerous reports on data and AI/ML best-practices from various perspectives and fields. Argonne, Oak Ridge, and Lawrence Berkeley national laboratories have recommended that computational science communities need AI/ML and massive datasets [212], including the suggestion that a lack of data is by far the largest threat to the dawn of strongly AI-enabled biology, advocating for advances in synthetic data generation, privacy-preserving ML, and data engineering. Leaders in the nuclear fusion community collaborate on a set of needs for AI to move the field forward [570], calling for an industry-wide "fusion data machine learning platform". NASA perspectives on big data systems for climate science [571, 572] cover topics such as data ETL (extract-transform-load) in massive multi-tier storage environments, and the need to discover and reuse data workflows. The National Science Foundation is pushing for domain-inspired AI architectures and optimization schemes to enable big-data driven discovery [573] across various domain-specific fields such as chemistry and materials sciences [574, 575], and agriculture [576]. Here we highlight several of the common themes across those reports, which are data themes shared with simulation intelligence. We also expand on them and clarify what is meant by concepts such as "ML-ready data" and "open data":

- **Interoperability** is one of the tenets of "FAIR" [577], guiding principles that define 15 ways in which data should be **f**indable, **a**ccessible, **i**nteroperable, and **r**eusable. In general these are rather straightforward: for example, to be findable data are assigned a globally unique and persistent identifier, to be reusable data are released with a clear and accessible data usage license, and so on. Although generally agreed upon, FAIR is rather high-level and warrants more specific details for the tenets. Double-clicking on interoperable is particularly important for the data-intensive applications of SI. *Data interoperability*, as it pertains to AI/ML and simulation sciences, addresses the ability of computational systems to exchange data with unambiguous, shared meaning and content. The medical ecosystem is unfortunately emblematic of the main data interoperability challenges [578] that stymie ML and scientific progress: incompatible standards, isolated and legacy databases, incompatible systems and proprietary software, inconsistent privacy and security concerns, and datasets that are inherently difficult to exchange, analyze, and interpret. Many of these interoperability challenges are a consequence of business incentives or legal needs. Other problems could be solved by at least a common ontology and a standardized representation of knowledge within this ontology – for example, the "SyBiOnt" ontology for applications in synthetic biology design [579], to bring together the collective wealth of existing biological information in databases and literature. We further suggest that such an ontology should integrate with a unanimous data lifecycle management framework in that field, where a main value-adds are proper semantic versioning for datasets and lingua franca amongst stakeholders [122]; more on such frameworks is mentioned in the readiness entry. When considering distributed computing and HPC (both highly important in scientific computing and simulation), these challenges can obstruct the sharing and interconnection of data such that scaling the experiments is infeasible.
- **Data readiness** at one time could have described the preparedness and availability of data for programmer queries and data science or analytics use. The set of concerns are far more complex and dynamic with ML. And further, scientific use-cases can impose stricter requirements. Data-readiness describes the successful execution

of the data tasks ahead of modeling: collating, processing, and curating data for the intended use(s). Challenges include poor and undocumented data collection practices, missing values, inconvenient storage mechanisms, intellectual property, security and privacy – notice the overlap with the interoperability issues above. To help overcome these challenges, several frameworks for data readiness have been developed [122, 580, 581]. These provide data handling structure and communication tools for ML practitioners, including required implementations such as data profiling and quality analyses as a key step before data enters a ML pipeline, and continuous testing for shifts in the data and environments of deployed models. These frameworks are highly relevant for SI practitioners and systems as well.

- **Provenance** in scientific workflows is paramount: a data product contains information about the process and data used to derive the data product, and provenance provides important documentation that is key to preserving the data, to determining the data's quality and authorship, and to reproduce as well as validate the results. These are all important requirements of the scientific process [582]. Several overviews on the topic point to methods for scientific data provenance as a means of reproducability [582] and integrity [583]. We suggest including *knowledge provenance* in this theme, referring to the audit trail, lineage, and pedigree of the computed information and results. In recent years the ML community has prioritized systems with tools for versioning datasets and models, which should be embraced by all in scientific computing and simulation, and extended to versioning of environments and testbeds.
- **Synthetic data** is paramount in scientific simulation and AI/ML. The primary use-case is data generation to make up for shortcomings in real data: availability, annotation, distribution (variance), representation of various modes and anomalies, computational constraints, and of course sample size. With the popularity of data-hungry deep learning in recent years, there have been many works on synthetic data generation for ML, largely in general computer vision and robotics [584, 585], and also scientific areas such as medical imaging and clinical records [586, 587, 588], social and economic systems [10, 589], particle physics [590] and astrophysics [591], and human cognition and behavior [226, 326, 327]. The validation of generated data against the real environment, and the reliability of subsequent models to perform in said environment, are of principal importance – Rankin et al. [592], for example, demonstrate the necessary analysis in the context of medical ML. A variety of statistical methods exist for quantifying the differences in data distributions [593] (and relatedly for dataset drifts [594]) In some use-cases, domain-specific metrics may be called for, imposing physics-informed or other environment constraints on the synthetic data [64].

  An overarching challenge is the integrated use of synthetic and real datasets. Techniques for "sim-to-real" transfer learning help ML models train synthetically and deploy with real data [593]. The AI/ML perspective of the AI and ML field is largely dominated by simulation testbeds (including video games) for robotics use-cases. However in scientific use-cases we encounter numerous scenarios where simulated data can be used as a catalyst in machine-based experiment understanding: generating synthetic diagnostics that enable the inference of quantities that are not directly measured in machine-based experiments – e.g., extracting meaningful physics from extremely noisy signals in particle physics, and automatically identifying important nuclear fusion plasma states and regimes for use in supervised learning. Humphreys et al. [570] refer to this as one means of "machine learning boosted diagnsotics", which also includes the systematic integration of multiple data sources that encompasses the next theme.
- **Data fusion** corresponds to the integration of information acquired from multiple sources (sensors, databases, manual trials, experiment sites, etc.). Many challenges exist that are highly dependent on the complexity of application environments and scales of sensors and systems. For example, data imperfections and inconsistencies due to noise in sensors and environments, the alignment or registration of data from different sensors or machines with different frames, and more discussed in the data fusion and ML survey by Meng et al. [595]. With open data, an ongoing challenge is finding true values from conflicting information when there are a large number of sources, among which some may copy from others [596]. Large-scale computer simulations of physical phenomena must be periodically corrected by using additional sources of information, including snapshots of reality and model forecasts. The fusion of observations into computer models is known as ***data assimilation*** and is extensively used in many areas, such as atmospheric prediction, seismology, and energy applications. Assimilating multiple sources of information into computer simulations helps maintain an accurate representation of the evolution of the physical phenomena and promotes realistic long-term prediction. Yet developing efficient and scalable assimilation algorithms capable of handling complex environments with nonlinear and non-Gaussian behavior is still an open research area, which is strongly aligned with several SI research directions we've discussed: optimal design of experiments and sensor placement, physics-infused optimization algorithms, and inverse problem solving.
- **Openness** – "open science" can be defined as a new approach to the scientific process based on cooperative work and new ways of diffusing knowledge by using digital technologies and new collaborative tools [597]. Hesitancy to share can come from several places including security restraints, concerns of intellectual property

(IP), and fear of being scooped by other researchers. Openness for data can be thought of as making data searchable, accessible, flexible, and reusable – open data is thus FAIR data. Open data also invites concerns with ethics and accountability, in addition to those prevalent in AI and scientific computing broadly. Please see Hutchinson et al. [598] and references therein for details. We further suggest that practical openness requires industry or domain specific standards that are agreed upon and adhered to – from appropriate metrics and storage formats, to procedures for gathering and labeling data – in order to resolve challenges in data acceptance, sharing, and longevity. These are also essential for multi-source databases, for instance the IDA (Image and Data Archive, ida.loni.usc.edu) with over 150 neuroimaging studies, most of which collect data from multiple clinical sites and over several years such that shared, agreed-upon data procedures are necessary for open data.

Data pipelines and platforms for SI Along with the SI motifs and their integrations that we've presented, the advancement of SI calls for cohesive data platforms that exemplify the above principles. The report on advancing nuclear fusion with ML research [570] urges that "the deployment of a Fusion Machine Learning Data Platform could in itself prove a transformational advance, dramatically increasing the ability of fusion science, mathematics, and computer science communities to combine their areas of expertise in accelerating the solution of fusion energy problems" [570]. Many fields in physical and life sciences could make the same assertion, including all the domains we've discussed in the context of SI. A recent example of such a platform is Microsoft's "Planetary Computer", an open-source cloud platform for multi-petabye global environmental data: planetarycomputer.microsoft.com. Largely an Earth observation catalogue and development hub, the datasets exemplify the above list of principles, and the ecosystem is designed to cultivate interdisciplinary and open science. The open-source code should eventually include pipelines for data processing and engineering, including modules for the principled use of real and synthetic data.

Hutchinson et al. [598] provide a thorough regimen for ML dataset lifecycles and supporting software infrastructure, while the design of SI data platforms should also consider how the new scientific workflows like Fig. 37 can have different requirements and challenges from ML in general. Consider that each phase of such a workflow may have several implicit dataflows, each executing a set of chained data transformations consuming and producing datasets; A challenge is these phases are often heterogeneous in a few ways: First, software, hardware, and development practices are inconsistent between collaborating people, teams, and organizations. Second, various fields call for domain-specific stuff such as specialized data types and structures. And mainly, most of the available software has grown through accretion, where code addition is driven by the need for producing domain science papers without adequate planning for code robustness and reliability. [599] The Department of Energy's 2019 AI for Science Report states "Today, the coupling of traditional modeling and simulation codes with AI capabilities is largely a one-off capability, replicated with each experiment. The frameworks, software, and data structures are distinct, and APIs do not exist that would enable even simple coupling of simulation and modeling codes with AI libraries and frameworks." The interdisciplinary nature of simulation intelligence, along with the many possible motif integrations, offer opportunity for SI data platforms to address these concerns.

The implementation and engineering of such data platforms is far from trivial [600, 601]. First consider that the curation and preparation of data are often the most time-consuming and manual tasks, which end up being disconnected and distributed in scientific endeavors. For the former, much manual and highly specialized work is performed by domain scientists to gather the relevant scientific data, and/or to use ML to achieve automated knowledge extraction from scientific data. Even then there is a significant gap between raw (or even cleaned) scientific data and useful data for consumption by ML and other parts of the workflow including simulators. Not to mention most scientific domains have specialized data formats that may require industry-specific software and domain-specific knowledge to inspect, visualize, and understand the data. See, for example, Simmhan et al. [601] discuss pipeline designs for reliable, scalable data ingestion in a distributed environment in order to support the Pan-STARRS repository, one of the largest digital surveys that accumulates 100TB of data annually to support 300 astronomers. Blamey et al. [602] also provide an informative walk through the challenges and implementations of cloud-native intelligent data pipelines for scientific data streams in life sciences. Deelman et al. [603] advocate for co-located (or *in situ*) computational workflows, to minimize inefficiencies with distributed computing. One motivation for co-located workflows is to minimize the cost of data movement by exploiting data locality by operating on data in place. Other supporting reasons include supporting human-in-the-loop interactive analysis, and capturing provenance for interactive data analysis and visualization (as well as any computational steering that results [v]). Data pipelines need to provide efficient, scalable, parallel, and

[v] *Computational steering* methods provide interactivity with an HPC program that is running remotely. It is most commonly used to alter parameters in a running program, receive real-time information from an application, and visualize the progress of an application. The idea is to save compute cycles by directing an application toward more productive areas of work, or at a minimum stopping unproductive work as early as possible. In the numerical simulation community it more specifically refers to the practice of interactively guiding a computational experiment into some region of interest.

resilient communication through flexible coupling of components with different data and resource needs, and utilize extreme-scale architectural characteristics such as deep memory/storage hierarchy and high core count.

Technologies in large scale data processing and engineering have flourished in the previous decades of "big data", and even more so with the accelerating adoption of AI and ML. It can be expected that SI will be a similar accelerant of data technologies and ecosystems. Large scale computing (i.e., peta-to-exa scale simulation) imposes significant challenges in engineering data pipelines [604, 605], in particular for the HPC community which we discuss in the next section.

Accelerated computing

High performance computing (HPC) plays a critical role in simulation systems and scientific computing. HPC systems are mainly used in domains with complex data-intensive applications, applying advanced mathematics and sophisticated parallel algorithms to solve large scale problems appearing in physics, medicine, mathematics, biology, and engineering [606]. Established supercomputer codes (where a supercomputer is system optimized for HPC applications) build on many years of research in parallelization [607, 608, 609] and numerical methods such as linear and nonlinear solvers, algorithms for continuous and discrete variables, automatic mesh refinements [610, 611] and others – for example, the PETSc (petsc.org) suite of data structures and routines for the scalable (parallel) solution of scientific applications modeled by partial differential equations [612]. Advances in ML and AI have impacted supercomputers in significant ways: enabling lower precision data types, leveraging tensor operations, and distributed machine learning frameworks. Further, it is becoming easier to solve traditional problems in a non-traditional way, leveraging neural networks. For instance, ML has proven highly successful for analyzing large-scale simulation data produced by supercomputers, such as extracting climate extremes [613], and towards efficient surrogate models or emulators based on HPC ensemble simulations [614]. One prominent example of a problem requiring massive computing resources is molecular dynamics (MD). "Anton" [615] is a family of supercomputers designed from scratch to solve precisely this one problem; recall from Fig. 5 that MD is one subset of the broader spatiotemporal space needed to simulate dynamics of our world. Recently, researchers at Argonne National Laboratory developed an AI-driven simulation framework for solving the same MD problem, yielding 50x speedup in time to solutions over the traditional HPC method [616]. And some work suggests these approaches need not be mutually exclusive: it has been shown in the context of computational fluid dynamics that traditional HPC workloads can be run alongside AI training to provide accelerated data-feed paths [617]. Co-locating workloads in this way may be necessary for petascale (approaching exascale) scientific simulations: Compute aside, supercomputers or large clusters (distributed compute nodes) are the only way to host some of the largest currently available models – as their memory requirements go into trillions of parameters, partitioning the model is the only way. In particular, the ensemble simulations can only be done in HPC. That said, a unified, two-way coupling between such large scale simulators and ML (which calls for data assimilation at each time step, or updating parameters and state variables) remains a significant challenge due to the complexity of the system's mathematical equations (which are often nonlinear) as well as the existing input-output (I/O) structures that are often built in a black-box way.

Many HPC experts suggest that the highest performing computing systems of the future will be specialized for the application at hand, customized in the hardware itself *and* in the algorithms and abstractions underlying the software.[vi] Both of these considerations – distributed compute and HW-SW specialization, not to mention heterogeneous systems incorporating a variety of specialized components – can be powerful but make application development more difficult [619]. At the hardware level, Fiore et al. [604] suggest new processors, memory hierarchy, and interconnects have to be taken into consideration when designing HPC systems to succeed existing petascale systems. Some works have stated that silicon photonic interconnects could provide large bandwidth densities at high-energy efficiencies to help solve some issues related to the increased parallelism and data intensity and movement [620, 621]. At the software level [599], compilers, libraries, and other middleware, should be adapted to these new platforms – perhaps in automated ways enabled by machine programming and program synthesis (e.g. in "Arbor" for neuronal network simulations on HPC [622]). At the applications level, algorithms, programming models, data access, analysis, visualization tools and overall scientific workflows [603], have to be redesigned when developing applications in petascale (potentially to exascale) computing infrastructures – for instance, HPC systems (and teams) would do well to adopt the cycles and tooling of the DevOps and newer MLOps [623] ecosystems, although unique characteristics related to HPC-level data storage and resource sharing will present new challenges. The interdisciplinary nature of these computing topics is

[vi] A different perspective can be drawn from the example of "Folding@home" [618], a distributed computing project for running protein dynamics simulations on networks of volunteers' personal computers that consist of heterogeneous hardware components. The project aims to help scientists develop new therapeutics for a variety of diseases by simulating the processes of protein folding and the movements of proteins at scale.

paramount: collaboration and synergy among the domain science, mathematics, computer science and engineering, modeling and simulation, etc., an optimal system and solution can be built. Like the integration of SI motifs and the broader SI stack, more comprehensive and holistic approaches are required to tackle the many complex real-world challenges.

An interesting and active use-case for HPC is quantum computing, where researchers use supercomputers to simulate quantum circuits [624, 625, 626, 627, 628] and quantum computers more generally. The invention of quantum computing was motivated [629] by the fact that quantum computers are exponentially (in the number of quantum bits, circuit depth, or both) expensive to simulate—this both gives QCs their power and also makes it very challenging to simulate even intermediate-scale QC [630]. Classical simulations of QCs are important in at least two areas: quantum-algorithm design (enabling empirical testing and verification of new quantum algorithms) and quantum-system design (enabling the simulation of both quantum hardware and layers of the quantum stack [631] above hardware but below algorithms). Accelerated and accurate simulation of quantum computers will play an important role in the development of practical, useful quantum computers. In parallel to the growth of conventional HPC acceleration of QC over the past several years, there has been an explosion of interest in the application of machine-learning tools to problems in quantum physics in general [632, 633, 634] and to the simulation of QC in particular [635]. We are already seeing SI approaches applied to the domain of QC. Another potential application of HPC to QC, which is essentially entirely separate from the simulation use case, is that of performing classical co-processing to aid in the execution of error-correction protocols for quantum hardware. Already there have been efforts to apply neural networks to help with error decoding [634], and it seems likely that quantum error correction will ultimately be implemented at scale using a combination of HPC and AI methods. Finally, the practical uses of QC typically involve both quantum and classical processing [636], with many use cases requiring very substantial amounts of classical computation that themselves warrant the use of HPC resources. As a concrete example, quantum-chemistry simulations are anticipated to involve a combination of both classical and QC computations [637] (which we've alluded to earlier in several motif examples).

As the field makes advances in the specific motifs and SI overall, there are several open questions to keep an eye on regarding computation:

- Will advances in the motifs lead to new breeds of algorithms that require radically new systems and/or architectures? Perhaps these can be implemented as conventional ASICs (Application-Specific Integrated Circuits), or will new substrates beyond CMOS electronics[638] be preferred?
- Will SI inverse design produce novel compute architectures and substrates?
- How will the open-endedness paradigm affect computing systems (memory allocation, load balancing, caching, etc.)?
- How will Machine Programming influence hardware-software co-design for simulation intelligence (and high-performance computing)?
- In addition to the traditional goals of performance and scalability (latency, throughput, etc.), energy efficiency, and reliability, how will the advancing technologies build abstractions to facilitate reasoning about correctness, security, and privacy [619]?
- Will SI-based surrogate modeling enable a landscape shift where computationally lightweight surrogates running on a small number of GPUs replace massive supercomputing workloads (particularly in physical and life sciences simulations)? What will be the metrics and benchmarks to quantify the tradeoffs between such orthogonal computation approaches?

To the first question, there has already been a resurgence of interest in novel hardware platforms for both simulation [639, 640] and machine learning [641, 642, 643, 644, 645, 646, 647]. Although CMOS platforms are still used essentially exclusively due to their performance and reliability, we anticipate that special-purpose hardware built on non-CMOS physical substrates may gain traction over the next decade. It remains to be seen if and how appropriate benchmarks are developed for comparing different compute substrates, which may ultimately be infeasible in a general, application-agnostic way. This echoes a significant benchmarking issue we discussed earlier in the context of physics-infused ML: It is customary that distributed training algorithms in HPC platforms are benchmarked using idealized neural network models and datasets [573], which does not impart any insights regarding the actual performance of these approaches in real deployment scenarios – i.e., when using domain-inspired AI architectures and optimization schemes for data-driven discovery in the context of realistic datasets (which are noisy, incomplete, and heterogeneous), and when using purpose-built HPC stacks that can have application-specific customizations up-and-down the layers we described above.

Domains for use-inspired research

Much of the work to develop the motif methods and their many integrations is *foundational SI research*: Understanding the mechanisms underlying thought and intelligent behavior and their implementation in machines. The main descriptions of each motif fall in this category, from the mathematics of causality to multi-agent cooperation. Pursuing foundational research in-sync with use-inspired research is essential, in SI and broadly in simulation, AI, and data technologies. *Use-inspired research* is basic research with uses for society in mind, thus a main driver of breakthrough innovations. Sometimes referred to as "Pasteur's quadrant"[vii], the many example motif applications we've discussed align with this division of research and development. The National Science Foundation similarly describes their AI Institutes as operating in cycles of foundational and use-inspired AI research: nsf.gov/cise/ai.jsp.

Table 2 provides a high-level, non-exhaustive look of the domains for use-inspired research per motif. Note many methods and use-cases arise from the integrations of motifs, as discussed throughout the text, which are not necessarily elucidated in this high-level table view.

Brain simulation We've discussed several neuroscience (and cognitive science) use-cases of SI motifs, notably with simulation-based inference and probabilistic programming (as well as "working memory" and inverse design). The intersections of neuroscience and simulation, including big data and machine learning, have had profound effects on the various fields for decades. The review by Fan & Markrum [648] provides a thorough overview that is out of scope for the current paper. The authors detail the last century of brain research, building up to today's supercomputer whole-brain models [649], with a focus on the recent phases of "simulation neuroscience" and the "multi-scale brain". The former describes a collaborative, integrated research setting where *in silico* and *in situ* experiments interplay to drive mathematical and predictive models (as well as produce hypotheses), all leveraging a wealth of biological datasets on a cohesive, open platform. The latter describes the need for causally linking molecular, cellular and synaptic interactions to brain function and behavior. We aim for SI to play critical roles in these phases of brain research for decades to come.

Simulations provide the crucial link between data generated by experimentalists and computational (as well as theoretical) neuroscience models; comparing predictions derived from such simulations with experimental data will allow systematic testing and refinement of models, and also inform theory in ways that will directly influence neuro-inspired AI (e.g., as in neural circuitry models for computer vision [319], working memory [650, 651], predictive coding [652, 653], and others such as spiking NNs [654], language of thought [655], and neuromorphic computing [656]) – see Refs. [657, 658, 659] for more discussion.

Discussion

In addition to the nine SI motifs, a number of advances in ML also provide encouraging directions for simulation-based sciences and ML, which we describe below as "honorable mention" methods in scientific AI and ML. We then close with discussion on the science of science: methods, breakthroughs, and the role for SI going forward.

Honorable mention motifs

Geometric and equivariant neural networks In various physics we encounter symmetries, or *equivariance*, defined as the property of being independent of the choice of reference frame. A function is equivariant to a transformation if applying that transformation to its inputs results in an equivalent transformation of its outputs. A special case of equivariance that has been exploited largely in machine learning is *invariance*, where any transformation of the inputs results in an identity transformation of the outputs: the architectures of CNNs and RNNs make them invariant to image translations and sequence position shifts, respectively. Thus neural networks with equivarience properties are an intriguing direction of research. Further, these methods share maths with category theory, which may help define equivariant architectures beyond CNN and RNN. To this end, Cohen & Welling [660] generalize CNNs to G-CNNs, where generalized-convolution replace the position shifting in a discrete convolution operation with a general group of transformations (like rotations and reflections, in addition to translations). More recent developments by Cohen et al. [661] extend neural network equivariance beyond group symmetries, developing neural networks that consume data on a general manifold dependent only on the intrinsic geometry of the manifold.

[vii] *Pasteur's quadrant* is a classification of scientific research projects that seek fundamental understanding of scientific problems, while also having immediate use for society. Louis Pasteur's research is thought to exemplify this type of method, which bridges the gap between "basic" and "applied" research.

	Machine programming	Differentiable programming	Probabilistic programming	Multi-physics multi-scale	Simulation-based inference	Surrogate modeling, emulation	Causal reasoning	Agent-based modeling	Open-endedness
HEPhysics - theoretical	-	✓	✓	✓	✓	-	✓	-	✓
HEPhysics - experimental	✓	✓	✓	✓	✓	✓	✓	-	-
Complexity	-	-	-	✓	-	-	✓	✓	✓
Synthetic biology	✓	✓	✓	✓	✓	✓	✓	-	-
Chemistry	-	✓	✓	✓	✓	✓	-	-	-
Materials	-	✓	✓	✓	✓	✓	-	-	-
Medicine	-	✓	✓	✓	✓	✓	✓	-	-
Systems biology	-	✓	✓	✓	-	✓	✓	✓	✓
Neuro and Cog sciences	✓	✓	✓	✓	✓	✓	✓	✓	✓
Energy - nuclear fission & fusion	-	✓	✓	✓	✓	✓	-	-	✓
Energy - materials & storage	-	✓	✓	✓	✓	✓	-	-	✓
Manufacturing	✓	✓	-	✓	-	✓	-	-	✓
Energy systems	-	-	-	✓	-	✓	✓	✓	✓
Transportation & infrastructure	-	-	-	✓	-	✓	✓	✓	✓
Agriculture	-	✓	-	✓	-	✓	✓	✓	✓
Ecology	-	✓	✓	✓	✓	✓	✓	✓	✓
Socioeconomics & markets	-	✓	-	✓	-	-	✓	✓	✓
Finance	✓	✓	✓	-	-	-	✓	✓	✓
Geopolitics	-	-	-	✓	-	-	✓	✓	✓
Defense	✓	✓	✓	✓	✓	✓	✓	✓	✓
Climate	-	✓	-	✓	✓	✓	✓	✓	✓
Earth systems	-	✓	-	✓	✓	✓	✓	✓	✓
Astrophysics & cosmology	-	✓	✓	✓	✓	✓	✓	-	✓

Table 2: The motifs can be applied for science and intelligence workflows in diverse domains and interdisciplinary use-cases. Here we itemize some main application areas (approximately ordered from smallest physical scales to largest). Note many methods and use-cases arise from the integrations of motifs (as discussed throughout the text), which are not necessarily apparent in this table view. Some useful information on the listed domains: *HEP* stands for high-energy physics (also called particle physics); *Complexity* includes systems and intelligence as defined by the Santa Fe Institute at *What is complex systems science?* and *Complex intelligence: natural, artificial, collective*, respectively; *Manufacturing* notably includes ML-based design of sensors and chips; *Earth systems* includes oceans, land, air, and near space (see earthdna.org).

Graph-structured data, one of the more prevalent types of data used in ML, presents many opportunities and challenges for exploiting symmetries. **Graph neural networks (GNNs)** are an extremely active area in the ML community for learning and predicting on flexible, graph-based inputs, with many applications in chemistry, material sciences, medicine, and others [662, 663]. Because of their scalability to large graphs, graph convolutional neural networks or message passing networks are widely used [664]. Yet these networks, which pass messages along the edges of the graph and aggregate them in a permutation invariant way, are fundamentally limited in their expressivity [665]. Equivariant graph networks are more expressive [666], thanks to the generalization properties (rooted in category theory). Recent work from Haan et al. [664] leverage category theory to derive a more expressive class of "natural graph networks" that can be used describe maximally flexible global and local graph networks – some of this relationship is discussed in [667].

Specializing GNN architectures for the task at hand has delivered state-of-art results in areas like drug discovery [668]. One can view such architecture specialities as inductive biases, specifically on the structure of the input (i.e., the input's dimensions are related such that they form a graph structure). This has taken shape as a class called ***Geometric Deep Learning***, where encoding the symmetries present in the problem domain are required for building neural network

models of geometric objects. Said another way, this direction of work aims to generalize equivariant and invariant neural network operators to non-Euclidean domains (e.g. graphs and manifolds) [669]. Bronstein et al. [669] illustrate that convolution commutes with the Laplacian operator, and they use this insight to define both spatially and spectrally motivated generalizations of convolution for non-Euclidean domains. In particular, equivariant neural networks have architectures designed for vector-transforms under symmetry operations (on reference frames) [661]. By constraining NNs to be symmetric under these operations, we've seen successful applications in areas like molecular design [670] and quantum chemistry [61]. There is also recent exploration in tying these structural constraints to classes of structural casual models (SCMs) that we discussed earlier, which could leade to theoretical foundations of how GNNs can be used to perform causal computations [671]. This is a robust area of ML and applied maths research – the interested reader can explore [672] for thorough overview, and [673] for theoretical foundations of group equivariance and NNs.

Learning mesh-based simulators Software for modeling of physical bodies and systems typically use mesh-based representations for the many geometries of surfaces and volumes. Meshes are used to solve underlying differential equations in the context of finite element analysis (FEA), computational fluid dynamics (CFD), and related statics and dynamics solvers. These present significant computational bottlenecks, which are weakly mitigated by adaptive meshes that are finer in areas that need high precision and coarser otherwise. New approaches from the machine learning community aim to alleviate such bottlenecks by learning the dynamics of deforming surfaces and volumes rather than the expensive discretization schemes [674] – that is, machine learned emulators for CFD and FEA. Pfaff et al. [675] develop a GNN model that can be trained to pass messages on a mesh graph and to adapt the mesh discretization during forward simulation. They show early results on common physical dynamics simulations of deforming plate, cloth in wind field, and flow are cylinders and airfoils. Similarly using learned message passing to compute dynamics, Sanchez-Gonzalez et al. [676] implement another GNN-based approach for efficiently simulating particle-based systems (and they suggest the same approach may apply to data represented as meshes as well). Both works point to the value of differentiable simulators, providing gradients that can be used for design optimization, optimal control tasks, and inverse problems.

Differentiable rendering Algorithms from the computer graphics community can have intriguing implications for ML, for instance for the generation of synthetic training data for computer vision [677, 678, 679] and the class of methods referred to as "vision as inverse graphics" [680]. Rendering in computer graphics is the process of generating images of 3D scenes defined by geometry, textures, materials, lighting, and the properties and positions of one or more cameras. Rendering is a complex process and its differentiation is not uniquely defined, yet differentiable rendering (DR) techniques based on autodiff can tackle such an integration for end-to-end optimization by obtaining useful gradients [681]. A driving interest from the ML perspective is towards computer vision models (largely deep neural networks and PGMs) that understand 3D information from 2D image data. One expectations is the improvement in vision applications while reducing the need for expensive 3D data collection and annotation. One general approach is an optimization pipeline with a differentiable renderer, where the gradients of an objective function with respect to the scene parameters and known ground-truth are calculated. Another common approach is self-supervision based on differentiable rendering, where the supervision signal is provided in the form of image evidence and the neural network is updated by backpropagating the error between the image and the rendering output. See Kato et al. [681] for an overview of DR algorithms, data representations, evaluation metrics, and best practices.

Normalizing and Manifold Flows Although very expressive generative models, a main limitation of the popular VAE and GAN approaches is intractable likelihoods (probability densities) which make inference challenging. They essentially represent a lower-dimensional data manifold that is embedded in the data space, and learn a mapping between the spaces. A class of generative models called *normalizing flows* [366, 367, 682] are based on a latent space with the same dimensionality as the data space and a diffeomorphism; their tractable density permeates the full data space and is not restricted to a lower-dimensional surface [683]. Normalizing flows describe a general mechanism for defining expressive probability distributions by transforming a simple initial density into a more complex one by applying a sequence of invertible (bijective) transformations [366, 367], can provide a richer, potentially more multi-modal distributions for probabilistic modeling. In theory the expressiveness is unlimited: Papamakarios et al. [367] show that for any pair of well-behaved distributions $p_x(\mathrm{x})$ and $p_u(mathrmu)$ (the target and base, respectively), there exists a diffeomorphism that can turn $p_u(\mathrm{u})$ into $p_x(\mathrm{x})$; x and u must have the same dimensionality and every sub-transformation along the way must preserve dimensionality. With distributions of larger dimensions there is larger computational cost per sub-transform; various classes of composable transformations exist, such as residual and radial flows, with known computational complexities. Multi-scale architectures can be used mitigate the growing complexity by clamping most dimensions in the chain of transforms, thus only applying all steps to a small subset of dimensions, which is less costly than applying all steps to all dimensions [684]. The same approach can be utilized for multi-scale modeling, which is a natural modeling choice in granular data settings such as pixels and waveforms [367], and in many scientific environments as we've discussed. A class of flows particularly relevant to SI may be *continuous-time flows*,

where the transformation from noise u to data x is described by an ordinary differential equation (rather than a series of discrete steps) [685, 686, 687]. There are also *equivariant manifold flows* [688] which aim to learn symmetry-invariant distributions on arbitrary manifolds, to overcome issues with the main generalized manifold-modeling approaches (both Euclidean and Riemannian). Much like the other equivariant methods we've discussed, we need to properly model symmetries for many applications in the natural sciences – e.g., symmetry requirements arise when sampling coupled particle systems in physical chemistry [689], or sampling for $SU(N)$ lattice gauge theories in theoretical physics in order to enable the construction of model architectures that respect the symmetries of the Standard Model of particle and nuclear physics and other physical theories [690]. For a thorough review of normalizing flows for probabilistic modeling and inference, refer to Papamakarios et al. [367].

Brehmer & Cranmer [683] highlight a limitation: Normalizing flows may be unsuited to data that do not populate the full ambient data space they natively reside in, and can learn a smeared-out version rather than the relatively higher dimensional manifold exactly (illustrated in Fig. 38). They introduce *manifold-learning flows (M-flows)*: normalizing flows based on an injective, invertible map from a lower dimensional latent space to the data space. The key novelty is that M-flows simultaneously learn the shape of the data manifold, provide a tractable bijective chart, and learn a probability density over the manifold, thus more faithfully representing input data that may be off manifold (in ambient data space). M-flows describe a new type of generative model that combines aspects of normalizing flows, autoencoders, and GANs; flows that simultaneously learn the data manifold and a tractable density over it may help us to unify generative and inference tasks in a way that is tailored to the structure of the data. In many scientific cases, domain knowledge allows for exact statements about the dimensionality of the data manifold, and M-flows can be a particularly powerful tool in a simulation-based inference setting [1].

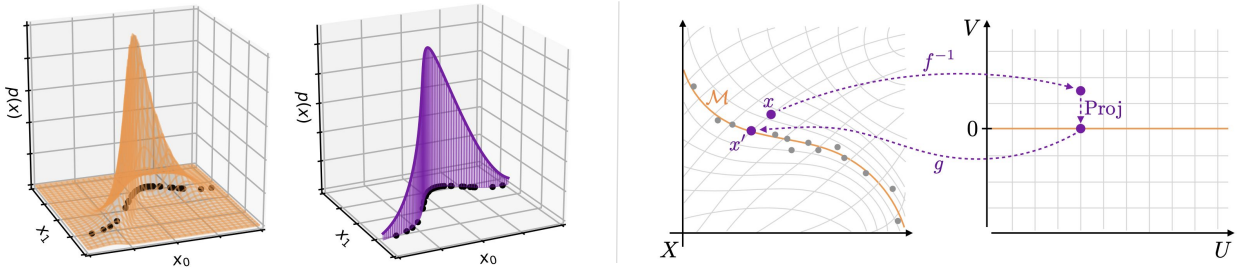

Figure 38: (Left) Sketch of how a standard normalizing flow in the ambient data space (left, orange surface) and an M-flow (right, purple) both model the same data (black dots). (Right) Sketch of how an M-flow evaluates arbitrary points on or off the learned manifold: First we show the data space X with data samples (grey) and the embedded manifold M (orange). Then the latent space UV is shown. In purple we sketch the evaluation of a data point x including its transformation to the latent space, the projection onto the manifold coordinates, and the transformation back to the manifold. (Figures reproduced from [683])

AlphaFold The recent landmark results from DeepMind's AlphaFold [358] on the Critical Assessment of Techniques for Protein Structure Prediction (CASP) competition showed dominance over the field for state-of-art in three-dimensional protein structure modeling.[viii] AlphaFold's approach to computational protein structure prediction is the first to achieve near-experimental accuracy in a majority of cases. Overall there were several deep learning innovations to assemble the network purpose-built for protein structures [358]: a new architecture to jointly embed multiple sequence alignments and pairwise features, a new output representation and associated loss that enable accurate end-to-end structure prediction, a new equivariant attention architecture, and others including curated loss functions. While the success of AlphaFold is undeniable, questions remain as to how much of the AlphaFold modeling framework, and knowledge about it, can be adapted to solve other, similar fundamental challenges in molecular biology, such as RNA folding and chromatin packing – how much effort would be needed to customize the tool to other areas to make a fresh contribution to them, and how to evaluate the contribution of the AI tool itself versus the human in the loop at most stages? A more philosophical outcome of AlphaFold is the implication that a scientific challenge can be considered partially or totally solved even if human scientists are unable to understand or gain any new knowledge from the solution, thus leaving open the question of whether the problem has really been solved. It will be intriguing to explore this perspective and the related area of AI-anthropomorphism [691] as more scientific, mathematical, and engineering challenges are overcome by various types of machine intelligence.

[viii] See "It will change everything: DeepMind's AI makes gigantic leap in solving protein structures" in Nature News, and "AlphaFold: a solution to a 50-year-old grand challenge in biology" from DeepMind.

Quantum computing Quantum computing [636] has strong connections with both simulation and machine learning. With the recent experimental demonstration of quantum computational supremacy [692] and rapid progress in the development of noisy intermediate-scale quantum (NISQ) machines [630] and algorithms[693], there is optimism that quantum computers might even deliver an advantage in tasks involving the simulation of quantum systems in the foreseeable future. In the longer term, once large-scale, fault-tolerant quantum computation is achieved, quantum simulation is widely expected to be a transformative use case [694, 695, 696].

There also exists a large body of work investigating diverse ways in which quantum computers might be used to perform machine-learning tasks – the subfield of *quantum machine learning (QML)* [697, 698]. QML methods exist both for tasks involving classical data as well as for tasks involving quantum data, although there is a general expectation that it will be easier to obtain an advantage with tasks on quantum data. The notion of differentiable programming has also been extended to quantum computing, first as part of QML, but now also applied to other areas of quantum algorithms [696, 699, 700, 701]. Since quantum algorithms – including quantum-simulation algorithms, but also algorithms simulating classical systems on quantum computers – can generate quantum data, there is a natural point of connection between simulation on quantum computers and QML for quantum data, and one might expect even closer interplay between simulation and machine learning methods in quantum computing, as we have seen develop in classical computing, and which this paper describes the merger of.

Conclusion

Donald Braben, a pioneer in innovation practices and the culture of scientific research, defines *transformative research* as "research that sets out radically to change the way we think about an important subject" [702]. Advancing the Nine Motifs and realizing synergistic artificial intelligence and simulation sciences can provide the requisite tools for transformative research across domains.

Yet simulation in conjunction with artificial intelligence does not necessarily yield solutions. Rather, when integrated in the ways we've described, it enables intelligent navigation of complex, high-dimensional spaces of possible solutions: bounding the space helps to guide the user to optimal solutions, and, in many cases, expanding the space to new territories can enable solutions beyond existing knowledge or human priors (Fig. 29).

Unlike current statistical machine learning approaches, simulation-based approaches can better deal with cause-effect relationships to help find first principles and the generating mechanisms required to make progress in areas of scientific discovery. Whether this is for problems in inverse design of synthetic biology compounds, predicting the results of interventions in social systems, or optimizing experimental design in high-energy physics, the Nine Motifs of SI explored here enable one to intelligently and tractably navigate the search space of possibilities.

This is why we put forth the concept of new SI-based scientific methods. While the traditional scientific method can be described as *hypothesis-driven deduction*, 21st century advances in big data brought about a major shift affecting nearly all fields, opening the door for a second dimension of the scientific method: *data-mining-inspired induction* [703]. More recently, machine learning along with high performance computing shaped this inductive reasoning class of scientific method to lean on the patterns and predictions from data-driven models.

It is clear the traditional scientific method provided a useful step-by-step guide for knowledge gathering, but this guide has also fundamentally shaped how humans think about the exploration of nature and scientific discovery. That is, presented with a new research question, scientists have been trained to think immediately in terms of hypotheses and alternatives, and how to practically implement controlled tests [703]. While effective for centuries, the process is also very slow, and subjective in the sense that it is driven by the scientist's ingenuity as well as bias – this bias has sometimes been a hindrance to necessary paradigm shifts [704]. It is possible that additional dimensions (or classes) of scientific methods can shift this classical regime to additional modes of knowledge gathering, which is what we aim to motivate in this work with simulation intelligence.

One can expect significant advances in science and intelligence with the development and adoption of these modes, but it is also possible that additional ethical and societal implications will arise. Automation bias and algorithm aversion are concerns we raised in the context of human-machine teaming, and can appear in varied ways when exploring new dimensions of scientific methods with simulation and artificial intelligence technologies. High costs associated with data collection, data access, and computing power are already problematic in today's society, particularly in the creation and exacerbation of inequalities in research output between state and private institutions as well as between developed and developing nations. Not to mention the ethics and safety challenges we encounter with AI in high-stakes settings, such as biomedical and social research, are similarly relevant with SI-based methods in these settings; performance

variability of healthcare and sociopolitical applications across subsets of the population are already a challenge of conventional data-driven methods, and increasing the reliance on such methods at the research stage might exacerbate these issues further.

The continued blending of data-driven, AI, simulation, and large-scale computing workflows is a promising path towards major progress and the emergence of a new scientific methodology [705]. It is not unreasonable to expect that the SI motifs and the integrations we've described can broaden the scientific method in general, with reliable and efficient hypothesis–simulation–analysis cycles [212], including machine-driven hypothesis generation for highly autonomous science. Each of these motifs, particularly in combinations described by the SI stack, allows this process to happen faster and more effectively – from probabilistic programming decoupling modeling and inference, to surrogate modeling allowing for rapid and powerful simulation of systems when detailed mechanistic modeling is infeasible, to simulation-based causal discovery deriving the true mechanisms of disease processes, to physics-infused learning deriving the governing equations of unknown physics, and so on. The SI driven scientific methods can help practitioners produce optimal and novel solutions across domains and use-cases in science and intelligence, and provide an essential step towards the *Nobel Turing Challenge*, creating the machine intelligence engine for scientific discovery [228].

References

[1] Kyle Cranmer, Johann Brehmer, and Gilles Louppe. The frontier of simulation-based inference. *Proceedings of the National Academy of Sciences*, 117(48):30055–30062, 2020. ISSN 0027-8424. doi: 10.1073/pnas.1912789117. URL https://www.pnas.org/content/117/48/30055.

[2] Robert Vautard, Andreas Gobiet, Daniela Jacob, Michal Belda, Augustin Colette, Michel Déqué, Jesús Fernández, Markel García-Díez, Klaus Goergen, Ivan Güttler, et al. The simulation of european heat waves from an ensemble of regional climate models within the euro-cordex project. *Climate Dynamics*, 41(9-10):2555–2575, 2013.

[3] John P. Dunne, Jasmin G. John, Elena Shevliakova, Ronald J. Stouffer, John P. Krasting, Sergey L. Malyshev, P. C. D. Milly, Lori T. Sentman, Alistair J. Adcroft, William Cooke, Krista A. Dunne, Stephen M. Griffies, Robert W. Hallberg, Matthew J. Harrison, Hiram Levy, Andrew T. Wittenberg, Peter J. Phillips, and Niki Zadeh. Gfdl's esm2 global coupled climate?carbon earth system models. part ii: Carbon system formulation and baseline simulation characteristics. *Journal of Climate*, 26(7):2247 – 2267, 2013. doi: 10.1175/JCLI-D-12-00150.1. URL https://journals.ametsoc.org/view/journals/clim/26/7/jcli-d-12-00150.1.xml.

[4] M. J. Roberts, P. L. Vidale, C. Senior, H. T. Hewitt, C. Bates, S. Berthou, P. Chang, H. M. Christensen, S. Danilov, M.-E. Demory, S. M. Griffies, R. Haarsma, T. Jung, G. Martin, S. Minobe, T. Ringler, M. Satoh, R. Schiemann, E. Scoccimarro, G. Stephens, and M. F. Wehner. The benefits of global high resolution for climate simulation: Process understanding and the enabling of stakeholder decisions at the regional scale. *Bulletin of the American Meteorological Society*, 99(11):2341 – 2359, 2018. doi: 10.1175/BAMS-D-15-00320.1. URL https://journals.ametsoc.org/view/journals/bams/99/11/bams-d-15-00320.1.xml.

[5] Joseph O Dada and Pedro Mendes. Multi-scale modelling and simulation in systems biology. *Integrative Biology*, 3(2):86–96, 2011.

[6] Ron O Dror, Robert M Dirks, JP Grossman, Huafeng Xu, and David E Shaw. Biomolecular simulation: a computational microscope for molecular biology. *Annual review of biophysics*, 41:429–452, 2012.

[7] Yasmina Elshafei, Matthew Tonts, M Sivapalan, and MR Hipsey. Sensitivity of emergent sociohydrologic dynamics to internal system properties and external sociopolitical factors: Implications for water management. *Water Resources Research*, 52(6): 4944–4966, 2016.

[8] Jacqueline M Hamilton, David J Maddison, and Richard SJ Tol. Climate change and international tourism: a simulation study. *Global environmental change*, 15(3):253–266, 2005.

[9] Ali Ramadhan, J. Marshall, A. Souza, G. Wagner, Manvitha Ponnapati, and Christopher Rackauckas. Capturing missing physics in climate model parameterizations using neural differential equations. *arXiv: Atmospheric and Oceanic Physics*, 2020.

[10] Stephan Zheng, Alexander Trott, Sunil Srinivasa, Nikhil Naik, Melvin Gruesbeck, David Parkes, and R. Socher. The ai economist: Improving equality and productivity with ai-driven tax policies. *ArXiv*, abs/2004.13332, 2020.

[11] Phillip Colella. Defining Software Requirements for Scientific Computing, 2004.

[12] K. Asanović, R. Bodík, Bryan Catanzaro, Joseph Gebis, P. Husbands, K. Keutzer, D. Patterson, W. Plishker, J. Shalf, S. Williams, and K. Yelick. The landscape of parallel computing research: A view from berkeley. 2006.

[13] Erich L Kaltofen. The "seven dwarfs" of symbolic computation. In *Numerical and symbolic scientific computing*, pages 95–104. Springer, 2012.

[14] Michael H. Matthee and S.P. Levitt. Domain specific languages contextualized. In *SAICSIT '11*, 2011.

[15] Samuel J. Gershman. The rational analysis of memory. 2019.

[16] Tyler Cowen and Ben Southwood. Is the rate of scientific progress slowing down? *SSRN Electronic Journal*, 2019.

[17] Jay Bhattacharya and Mikko Packalen. Stagnation and scientific incentives. *PSN: Science & Technology (Topic)*, 2020.

[18] D. Kochkov, J. A. Smith, Ayya Alieva, Qifeng Wang, M. Brenner, and Stephan Hoyer. Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences of the United States of America*, 118, 2021.

[19] Oliver Hennigh, S. Narasimhan, M. Nabian, A. Subramaniam, K. Tangsali, M. Rietmann, J. Ferrandis, Wonmin Byeon, Zhiwei Fang, and Sanjay Choudhry. Nvidia simnet: an ai-accelerated multi-physics simulation framework. *ArXiv*, abs/2012.07938, 2020.

[20] C. Hunt, A. Erdemir, W. Lytton, F. M. Gabhann, E. Sander, M. Transtrum, and Lealem Mulugeta. The spectrum of mechanism-oriented models and methods for explanations of biological phenomena. 2018.

[21] M. Alber, A. Buganza Tepole, W. Cannon, S. De, S. Dura-Bernal, K. Garikipati, G. Karniadakis, W. Lytton, P. Perdikaris, L. Petzold, and E. Kuhl. Integrating machine learning and multiscale modeling—perspectives, challenges, and opportunities in the biological, biomedical, and behavioral sciences. *NPJ Digital Medicine*, 2, 2019.

[22] P. Perdikaris and G. Karniadakis. Model inversion via multi-fidelity bayesian optimization: a new paradigm for parameter estimation in haemodynamics, and beyond. *Journal of The Royal Society Interface*, 13, 2016.

[23] F. S. Costabal, P. Perdikaris, E. Kuhl, and D. Hurtado. Multi-fidelity classification using gaussian processes: accelerating the prediction of large-scale computational models. *ArXiv*, abs/1905.03406, 2019.

[24] G. Karniadakis, I. Kevrekidis, Lu Lu, P. Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Rev. Phys.*, 2021.

[25] Shengze Cai, Zhiping Mao, Zhicheng Wang, Minglang Yin, and G. Karniadakis. Physics-informed neural networks (pinns) for fluid mechanics: A review. *ArXiv*, abs/2105.09506, 2021.

[26] C. Rasmussen and Christopher K. I. Williams. Gaussian processes for machine learning. 2006.

[27] Zoubin Ghahramani. Probabilistic machine learning and artificial intelligence. *Nature*, 521:452–459, 2015.

[28] M. Raissi, P. Perdikaris, and G. Karniadakis. Numerical gaussian processes for time-dependent and nonlinear partial differential equations. *ArXiv*, abs/1703.10230, 2018.

[29] M. Raissi and G. Karniadakis. Hidden physics models: Machine learning of nonlinear partial differential equations. *ArXiv*, abs/1708.00588, 2018.

[30] F. Lindgren, H. Rue, and J. Lindström. An explicit link between gaussian fields and gaussian markov random fields: the stochastic partial differential equation approach. *Journal of The Royal Statistical Society Series B-statistical Methodology*, 73: 423–498, 2011.

[31] V. Borovitskiy, Alexander Terenin, P. Mostowsky, and M. Deisenroth. Matern gaussian processes on riemannian manifolds. *ArXiv*, abs/2006.10160, 2020.

[32] A. Hanuka, X. Huang, J. Shtalenkova, D. Kennedy, A. Edelen, V. Lalchand, D. Ratner, and J. Duris. Physics-informed gaussian process for online optimization of particle accelerators. *ArXiv*, abs/2009.03566, 2020.

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

[34] Yinhao Zhu, N. Zabaras, P. Koutsourelakis, and P. Perdikaris. Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data. *ArXiv*, abs/1901.06314, 2019.

[35] M. Raissi, Zhicheng Wang, M. Triantafyllou, and G. Karniadakis. Deep learning of vortex-induced vibrations. *Journal of Fluid Mechanics*, 861:119 – 137, 2018.

[36] Dongkun Zhang, L. Lu, Ling Guo, and G. Karniadakis. Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems. *J. Comput. Phys.*, 397, 2019.

[37] X. Meng and G. Karniadakis. A composite neural network that learns from multi-fidelity data: Application to function approximation and inverse pde problems. *J. Comput. Phys.*, 401, 2020.

[38] Jens Berg and K. Nyström. A unified deep artificial neural network approach to partial differential equations in complex geometries. *Neurocomputing*, 317:28–41, 2018.

[39] Justin A. Sirignano and K. Spiliopoulos. Dgm: A deep learning algorithm for solving partial differential equations. *Journal of Computational Physics*, 375:1339–1364, 2018.

[40] Luning Sun, Han Gao, Shaowu Pan, and Jian-Xun Wang. Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data. *Computer Methods in Applied Mechanics and Engineering*, 361:112732, 2020.

[41] D. Koller and N. Friedman. Probabilistic graphical models - principles and techniques. 2009.

[42] E. J. Hall, Søren Taverniers, M. Katsoulakis, and D. Tartakovsky. Ginns: Graph-informed neural networks for multiscale physics. *J. Comput. Phys.*, 433, 2021.

[43] Philipp Hennig, Michael A. Osborne, and Mark A. Girolami. Probabilistic numerics and uncertainty in computations. *Proceedings. Mathematical, Physical, and Engineering Sciences / The Royal Society*, 471, 2015.

[44] L. Yang, X. Meng, and G. Karniadakis. B-pinns: Bayesian physics-informed neural networks for forward and inverse pde problems with noisy data. *J. Comput. Phys.*, 425:109913, 2021.

[45] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

[46] Rui Wang, K. Kashinath, M. Mustafa, A. Albert, and Rose Yu. Towards physics-informed deep learning for turbulent flow prediction. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2020.

[47] Zong-Yi Li, Nikola B. Kovachki, K. Azizzadenesheli, Burigede Liu, K. Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *ArXiv*, abs/2010.08895, 2020.

[48] M. Raissi, A. Yazdani, and G. Karniadakis. Hidden fluid mechanics: Learning velocity and pressure fields from flow visualizations. *Science*, 367:1026 – 1030, 2020.

[49] P. Perdikaris, L. Grinberg, and G. Karniadakis. Multiscale modeling and simulation of brain blood flow. *Physics of fluids*, 28 2:021304, 2016.

[50] S. Schoenholz and E. D. Cubuk. Jax md: A framework for differentiable physics. In *NeurIPS*, 2020.

[51] Keith T. Butler, Daniel W. Davies, Hugh M. Cartwright, Olexandr Isayev, and Aron Walsh. Machine learning for molecular and materials science. *Nature*, 559:547–555, 2018.

[52] Aron Walsh, Alexey A. Sokol, and Richard Catlow. Computational approaches to energy materials. 2013.

[53] Kurt Lejaeghere, Gustav Bihlmayer, Torbjörn Björkman, Peter Blaha, Stefan Blügel, Volker Blum, Damien Caliste, Ivano Eligio Castelli, Stewart J Clark, Andrea Dal Corso, Stefano de Gironcoli, Thierry Deutsch, J. K. Dewhurst, Igor Di Marco, Claudia Draxl, Marcin Dulak, Olle Eriksson, José A. Flores-Livas, Kevin F. Garrity, Luigi Genovese, Paolo Giannozzi, Matteo Giantomassi, Stefan Goedecker, Xavier Gonze, Oscar Grånäs, Eberhard K. U. Gross, Andris Gulans, François Gygi, Don R. Hamann, Philip Hasnip, N. A. W. Holzwarth, Diana Iuşan, Dominik Jochym, François Jollet, Daniel Jones, Georg Kresse, Klaus Koepernik, Emine Küçükbenli, Yaroslav O Kvashnin, Inka L. M. Locht, Sven Lubeck, Martijn Marsman, Nicola Marzari, Ulrike Nitzsche, Lars Nordström, Taisuke Ozaki, Lorenzo Paulatto, Chris J. Pickard, Ward Poelmans, Matt Probert, Keith Refson, Manuel Richter, Gian-Marco Rignanese, Santanu Saha, Matthias Scheffler, Martin Schlipf, Karlheinz Schwarz, Sangeeta Sharma, Francesca Tavazza, Patrik Thunström, Alexandre Tkatchenko, Marc Torrent, David Vanderbilt, Michiel J. van Setten, Veronique Van Speybroeck, John Michael Wills, Jonathan R. Yates, Guo-Xu Zhang, and Stefaan Cottenier. Reproducibility in density functional theory calculations of solids. *Science*, 351, 2016.

[54] Johannes Hachmann, Roberto Olivares-Amaya, Sule Atahan-Evrenk, Carlos Amador-Bedolla, Roel S. Sánchez-Carrera, Aryeh Gold-Parker, Leslie Vogt, Anna M. Brockway, and Alán Aspuru-Guzik. The harvard clean energy project: Large-scale computational screening and design of organic photovoltaics on the world community grid. *Journal of Physical Chemistry Letters*, 2:2241–2251, 2011.

[55] Anubhav Jain, Shyue Ping Ong, Geoffroy Hautier, Wei Chen, William Davidson Richards, Stephen T Dacek, Shreyas Cholia, Dan Gunter, D. Skinner, Gerbrand Ceder, and Kristin Aslaug Persson. Commentary: The materials project: A materials genome approach to accelerating materials innovation. *APL Materials*, 1:011002–011002, 2013.

[56] Camilo E. Calderon, Jose J Plata, Cormac Toher, Corey Oses, Ohad Levy, Marco Fornari, Amir Natan, Michael J. Mehl, Gus L. W. Hart, Marco Buongiorno Nardelli, and Stefano Curtarolo. The aflow standard for high-throughput materials science calculations. *Computational Materials Science*, 108:233–238, 2015.

[57] Anders S. Christensen, Sai Krishna Sirumalla, Zhuoran Qiao, Michael B. O'Connor, Daniel G. A. Smith, Feizhi Ding, Peter J. Bygrave, Anima Anandkumar, Matthew Welborn, Frederick R. Manby, and Thomas F. Miller. Orbnet denali: A machine learning potential for biological and organic chemistry with semi-empirical cost and dft accuracy. 2021.

[58] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. *ArXiv*, abs/1704.01212, 2017.

[59] Zhuoran Qiao, Matthew Welborn, Animashree Anandkumar, Frederick R. Manby, and Thomas F. Miller. Orbnet: Deep learning for quantum chemistry using symmetry-adapted atomic-orbital features. *The Journal of chemical physics*, 153 12, 2020.

[60] Gregor N. C. Simm, Robert Pinsler, Gábor Csányi, and José Miguel Hernández-Lobato. Symmetry-aware actor-critic for 3d molecular design. *ArXiv*, abs/2011.12747, 2021.

[61] Zhuoran Qiao, Anders S. Christensen, Frederick R. Manby, Matthew Welborn, Anima Anandkumar, and Thomas F. Miller. Unite: Unitary n-body tensor equivariant network with applications to quantum chemistry. *ArXiv*, abs/2105.14655, 2021.

[62] R. Basri, D. Jacobs, Yoni Kasten, and S. Kritchman. The convergence rate of neural networks for learned functions of different frequencies. In *NeurIPS*, 2019.

[63] Sifan Wang, Xinling Yu, and P. Perdikaris. When and why pinns fail to train: A neural tangent kernel perspective. *ArXiv*, abs/2007.14527, 2020.

[64] Björn Lütjens, Brandon Leshchinskiy, Christian Requena-Mesa, F. Chishtie, Natalia Díaz Rodríguez, Océane Boulais, Aruna Sankaranarayanan, A. Piña, Y. Gal, Chedy Raïssi, Alexander Lavin, and Dava Newman. Physically-consistent generative adversarial networks for coastal flood visualization. *ArXiv*, abs/2104.04785, 2021.

[65] Pantelis R. Vlachas, Jaideep Pathak, B. Hunt, T. Sapsis, M. Girvan, E. Ott, and P. Koumoutsakos. Backpropagation algorithms and reservoir computing in recurrent neural networks for the forecasting of complex spatiotemporal dynamics. *Neural networks : the official journal of the International Neural Network Society*, 126:191–217, 2020.

[66] Christopher Rackauckas, Y. Ma, Julius Martensen, Collin Warner, K. Zubov, R. Supekar, Dominic J. Skinner, and Ali Ramadhan. Universal differential equations for scientific machine learning. *ArXiv*, abs/2001.04385, 2020.

[67] D. Pfau, J. Spencer, A. G. D. G. Matthews, and W. Foulkes. Ab-initio solution of the many-electron schrödinger equation with deep neural networks. *ArXiv*, abs/1909.02487, 2019.

[68] Christopher Rackauckas, A. Edelman, K. Fischer, Mike Innes, Elliot Saba, Viral B. Shah, and Will Tebbutt. Generalized physics-informed learning through language-wide differentiable programming. In *AAAI Spring Symposium: MLPS*, 2020.

[69] E. Goldstein and G. Coco. Machine learning components in deterministic models: hybrid synergy in the age of data. *Frontiers in Environmental Science*, 3, 2015.

[70] M. Kennedy and A. O'Hagan. Bayesian calibration of computer models. *Journal of The Royal Statistical Society Series B-statistical Methodology*, 63:425–464, 2001.

[71] M. Titsias. Variational learning of inducing variables in sparse gaussian processes. In *AISTATS*, 2009.

[72] J. Hensman, Nicoló Fusi, and Neil Lawrence. Gaussian processes for big data. *ArXiv*, abs/1309.6835, 2013.

[73] A. G. Wilson, Christoph Dann, and H. Nickisch. Thoughts on massively scalable gaussian processes. *ArXiv*, abs/1511.01870, 2015.

[74] Pavel Izmailov, Alexander Novikov, and D. Kropotov. Scalable gaussian processes with billions of inducing inputs via tensor train decomposition. In *AISTATS*, 2018.

[75] Hongzhou Lin and S. Jegelka. Resnet with one-neuron hidden layers is a universal approximator. In *NeurIPS*, 2018.

[76] D. Winkler and T. Le. Performance of deep and shallow neural networks, the universal approximation theorem, activity cliffs, and qsar. *Molecular Informatics*, 36, 2017.

[77] I. Goodfellow, Yoshua Bengio, and Aaron C. Courville. Deep learning. *Nature*, 521:436–444, 2015.

[78] T. Chen, Yulia Rubanova, J. Bettencourt, and D. Duvenaud. Neural ordinary differential equations. In *NeurIPS*, 2018.

[79] Hanshu Yan, Jiawei Du, V. Tan, and Jiashi Feng. On robustness of neural ordinary differential equations. *ArXiv*, abs/1910.05513, 2020.

[80] B. Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and N. D. Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104:148–175, 2016.

[81] Andreas Krause and Carlos Guestrin. Nonmyopic active learning of gaussian processes: an exploration-exploitation approach. In *ICML '07*, 2007.

[82] Ilias Bilionis and Nicholas Zabaras. Bayesian uncertainty propagation using gaussian processes. 2015.

[83] Matteo Aldeghi, Florian Hase, Riley J. Hickman, Isaac Tamblyn, and Alán Aspuru-Guzik. Golem: An algorithm for robust experiment and process optimization. *ArXiv*, abs/2103.03716, 2021.

[84] M. McIntire, D. Ratner, and S. Ermon. Sparse gaussian processes for bayesian optimization. In *UAI*, 2016.

[85] P. Frazier. A tutorial on bayesian optimization. *ArXiv*, abs/1807.02811, 2018.

[86] Jasper Snoek, H. Larochelle, and Ryan P. Adams. Practical bayesian optimization of machine learning algorithms. In *NIPS*, 2012.

[87] Ryan-Rhys Griffiths and José Miguel Hernández-Lobato. Constrained bayesian optimization for automatic chemical design using variational autoencoders. 2019.

[88] R. Gómez-Bombarelli, D. Duvenaud, José Miguel Hernández-Lobato, J. Aguilera-Iparraguirre, Timothy D. Hirzel, Ryan P. Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS Central Science*, 4:268 – 276, 2018.

[89] Yichi Zhang, D. Apley, and W. Chen. Bayesian optimization for materials design with mixed quantitative and qualitative variables. *Scientific Reports*, 10, 2020.

[90] Matthias Poloczek, P. Frazier, Rémi Lam, and K. Willcox. Advances in bayesian optimization with applications in aerospace engineering. 2018.

[91] D. Lindley. On a measure of the information provided by an experiment. *Annals of Mathematical Statistics*, 27:986–1005, 1956.

[92] K. Chaloner and I. Verdinelli. Bayesian experimental design: A review. *Statistical Science*, 10:273–304, 1995.

[93] Adam Foster, Desi R. Ivanova, I. Malik, and Tom Rainforth. Deep adaptive design: Amortizing sequential bayesian experimental design. In *ICML*, 2021.

[94] Jiaxin Zhang, Sirui Bi, and Guannan Zhang. A scalable gradient free method for bayesian experimental design with implicit models. In *International Conference on Artificial Intelligence and Statistics*, pages 3745–3753. PMLR, 2021.

[95] J. Vanlier, C. A. Tiemann, P. Hilbers, and N. Riel. A bayesian approach to targeted experiment design. *Bioinformatics*, 28: 1136 – 1142, 2012.

[96] Jiaying Lyu, Emily Curran, and Y. Ji. Bayesian adaptive design for finding the maximum tolerated sequence of doses in multicycle dose-finding clinical trials. 2018.

[97] S. Dushenko, K. Ambal, and R. McMichael. Sequential bayesian experiment design for optically detected magnetic resonance of nitrogen-vacancy centers. *Physical review applied*, 14 5, 2020.

[98] Jay I. Myung, Daniel R. Cavagnaro, and M. Pitt. A tutorial on adaptive design optimization. *Journal of mathematical psychology*, 57 3-4:53–67, 2013.

[99] Riley J. Hickman, Florian Hase, L. Roch, and Alán Aspuru-Guzik. Gemini: Dynamic bias correction for autonomous experimentation and molecular simulation. *ArXiv*, abs/2103.03391, 2021.

[100] Sergey Shirobokov, Vladislav Belavin, Michael Kagan, Andrey Ustyuzhanin, and Atilim Gunes Baydin. Black-box optimization with local generative surrogates. In *Advances in Neural Information Processing Systems 34 (NeurIPS)*, 2020.

[101] P. Bauer, P. Dueben, Torsten Hoefler, T. Quintino, T. Schulthess, and N. Wedi. The digital revolution of earth-system science. 2021.

[102] Peishi Jiang, Nis Meinert, Helga Jordão, Constantin Weisser, Simon J. Holgate, Alexander Lavin, Bjorn Lutjens, Dava Newman, Haruko M. Wainwright, Catherine Walker, and Patrick L. Barnard. Digital twin earth – coasts: Developing a fast and physics-informed surrogate model for coastal floods via neural operators. In *NeurIPS Workshop on Machine Learning and the Physical Sciences*, 2021.

[103] Gurvan Madec. Nemo ocean engine. 2008.

[104] Patrick L. Barnard, M. van Ormondt, Li H. Erikson, Jodi L. Eshleman, Cheryl J. Hapke, Peter Ruggiero, Peter N. Adams, and Amy C. Foxgrover. Development of the coastal storm modeling system (cosmos) for predicting the impact of storms on high-energy, active-margin coasts. *Natural Hazards*, 74:1095–1125, 2014.

[105] N. Friedman. Inferring cellular networks using probabilistic graphical models. *Science*, 303:799 – 805, 2004.

[106] N. Beerenwinkel, R. Schwarz, M. Gerstung, and F. Markowetz. Cancer evolution: Mathematical models and computational inference. *Systematic Biology*, 64:e1 – e25, 2015.

[107] M. Leifer and D. Poulin. Quantum graphical models and belief propagation. *Annals of Physics*, 323:1899–1946, 2008.

[108] S. Boixo, S. Isakov, V. Smelyanskiy, and H. Neven. Simulation of low-depth quantum circuits as complex undirected graphical models. *arXiv: Quantum Physics*, 2017.

[109] T. Elsken, J. H. Metzen, and F. Hutter. Neural architecture search: A survey. *ArXiv*, abs/1808.05377, 2019.

[110] Rushil Anirudh, Jayaraman J. Thiagarajan, P. Bremer, and B. Spears. Improved surrogates in inertial confinement fusion with manifold and cycle consistencies. *Proceedings of the National Academy of Sciences of the United States of America*, 117: 9741 – 9746, 2020.

[111] J. Galdón-Quiroga, M. Garcia-Munoz, K. McClements, M. Nocente, M. Hoelzl, A. Jacobsen, F. Orain, J. Rivero-Rodríguez, M. Salewski, L. Sanchis-Sanchez, W. Suttrop, and E. Viezzer. Beam-ion acceleration during edge localized modes in the asdex upgrade tokamak. *Physical review letters*, 121 2:025002, 2018.

[112] I. Tegen, D. Neubauer, S. Ferrachat, C. S. Drian, I. Bey, N. Schutgens, P. Stier, D. Watson-Parris, T. Stanelle, H. Schmidt, S. Rast, H. Kokkola, M. Schultz, S. Schroeder, N. Daskalakis, Stefan Barthel, B. Heinold, and U. Lohmann. The global aerosol–climate model echam6.3–ham2.3 – part 1: Aerosol evaluation. *Geoscientific Model Development*, 12:1643–1677, 2019.

[113] S. Khatiwala. A computational framework for simulation of biogeochemical tracers in the ocean. *Global Biogeochemical Cycles*, 21, 2007.

[114] Colin White, W. Neiswanger, and Yash Savani. Bananas: Bayesian optimization with neural architectures for neural architecture search. In *AAAI*, 2021.

[115] S. Brunton, J. Proctor, and N. Kutz. Sparse identification of nonlinear dynamics (sindy). *Bulletin of the American Physical Society*, 2016.

[116] N. Mangan, S. Brunton, J. Proctor, and J. Kutz. Inferring biological networks by sparse identification of nonlinear dynamics. *IEEE Transactions on Molecular, Biological and Multi-Scale Communications*, 2:52–63, 2016.

[117] Bhavana Bhadriraju, Abhinav Narasingam, and J. Kwon. Machine learning-based adaptive model identification of systems: Application to a chemical process. *Chemical Engineering Research & Design*, 152:372–383, 2019.

[118] F. Cichos, K. Gustavsson, B. Mehlig, and G. Volpe. Machine learning for active matter. *Nature Machine Intelligence*, 2: 94–103, 2020.

[119] S. Rudy, S. Brunton, J. Proctor, and J. Kutz. Data-driven discovery of partial differential equations. *Science Advances*, 3, 2017.

[120] H. Schaeffer. Learning partial differential equations via data discovery and sparse optimization. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 473, 2017.

[121] Zhao Chen, Y. Liu, and Hao Sun. Deep learning of physical laws from scarce data. *ArXiv*, abs/2005.03448, 2020.

[122] Alexander Lavin, Ciarán M. Gilligan-Lee, Alessya Visnjic, Siddha Ganju, Dava Newman, Sujoy Ganguly, Danny Lange, Atılım Güneş Baydin, Amit Sharma, Adam Gibson, Yarin Gal, Eric P. Xing, Chris Mattmann, and James Parr. Technology readiness levels for machine learning systems, 2021.

[123] M. Cranmer, Alvaro Sanchez-Gonzalez, Peter W. Battaglia, Rui Xu, Kyle Cranmer, David N. Spergel, and Shirley Ho. Discovering symbolic models from deep learning with inductive biases. *ArXiv*, abs/2006.11287, 2020.

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

[125] Silviu-Marian Udrescu, Andrew Yong-Yi Tan, Jiahai Feng, Orisvaldo Neto, Tailin Wu, and Max Tegmark. Ai feynman 2.0: Pareto-optimal symbolic regression exploiting graph modularity. *ArXiv*, abs/2006.10782, 2020.

[126] Giancarlo Gandolfo. Giuseppe palomba and the lotka-volterra equations. *RENDICONTI LINCEI*, 19:347–357, 2008.

[127] Henry Markram, Eilif Muller, Srikanth Ramaswamy, Michael W Reimann, Marwan Abdellah, Carlos Aguado Sanchez, Anastasia Ailamaki, Lidia Alonso-Nanclares, Nicolas Antille, Selim Arsever, et al. Reconstruction and simulation of neocortical microcircuitry. *Cell*, 163(2):456–492, 2015.

[128] Scott A. Sisson. *Handbook of Approximate Bayesian Computation*. Chapman and Hall/CRC, 2018. doi: 10.1201/9781315117195.

[129] Alvaro Tejero-Cantero, Jan Boelts, Michael Deistler, Jan-Matthis Lueckmann, Conor Durkan, Pedro J Gonçalves, David S Greenberg, and Jakob H Macke. sbi: A toolkit for simulation-based inference. *Journal of Open Source Software*, 5(52):2505, 2020.

[130] Jarno Lintusaari, Henri Vuollekoski, Antti Kangasraasio, Kusti Skytén, Marko Jarvenpaa, Pekka Marttinen, Michael U Gutmann, Aki Vehtari, Jukka Corander, and Samuel Kaski. Elfi: Engine for likelihood-free inference. *Journal of Machine Learning Research*, 19(16):1–7, 2018.

[131] Emmanuel Klinger, Dennis Rickert, and Jan Hasenauer. pyabc: distributed, likelihood-free inference. *Bioinformatics*, 34(20): 3591–3593, 2018.

[132] Jan-Matthis Lueckmann, Jan Boelts, David Greenberg, Pedro Goncalves, and Jakob Macke. Benchmarking simulation-based inference. In *International Conference on Artificial Intelligence and Statistics*, pages 343–351. PMLR, 2021.

[133] Pedro J Gonçalves, Jan-Matthis Lueckmann, Michael Deistler, Marcel Nonnenmacher, Kaan Öcal, Giacomo Bassetto, Chaitanya Chintaluri, William F Podlaski, Sara A Haddad, Tim P Vogels, et al. Training deep neural density estimators to identify mechanistic models of neural dynamics. *Elife*, 9:e56261, 2020.

[134] Donald B. Rubin. Bayesianly Justifiable and Relevant Frequency Calculations for the Applied Statistician. *The Annals of Statistics*, 12(4):1151–1172, 1984. ISSN 0090-5364. doi: 10.1214/aos/1176346785. URL https://doi.org/10.1214/aos/1176346785.

[135] Mark A. Beaumont, Wenyang Zhang, and David J. Balding. Approximate Bayesian computation in population genetics. *Genetics*, 162(4):2025–2035, 2002. ISSN 00166731. doi: 10.1111/j.1937-2817.2010.tb01236.x.

[136] Monte Carlo Methods of Inference for Implicit Statistical Models. In *Journal of the Royal Statistical Society: Series B (Methodological)*, volume 46, pages 193–212, 1984. doi: 10.1111/j.2517-6161.1984.tb01290.x.

[137] Christian P Robert, Jean-Marie Cornuet, Jean-Michel Marin, and Natesh S Pillai. Lack of confidence in approximate bayesian computation model choice. *Proceedings of the National Academy of Sciences*, 108(37):15112–15117, 2011.

[138] Bai Jiang, Tung-yu Wu, Charles Zheng, and Wing H Wong. Learning summary statistic for approximate bayesian computation via deep neural network. *Statistica Sinica*, pages 1595–1618, 2017.

[139] Yanzhi Chen, Dinghuai Zhang, Michael U Gutmann, Aaron Courville, and Zhanxing Zhu. Neural approximate sufficient statistics for implicit models. In *Ninth International Conference on Learning Representations 2021*, 2021.

[140] Paul Fearnhead and Dennis Prangle. Constructing summary statistics for approximate bayesian computation: semi-automatic approximate bayesian computation. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 74(3): 419–474, 2012.

[141] George Papamakarios, David C. Sterratt, and Iain Murray. Sequential Neural Likelihood: Fast Likelihood-free Inference with Autoregressive Flows. *International Conference on Artificial Intelligence and Statistics*, 2018. URL http://arxiv.org/abs/1805.07226.

[142] Jan-Matthis Lueckmann, Giacomo Bassetto, Theofanis Karaletsos, and Jakob H. Macke. Likelihood-free inference with emulator networks. In Francisco Ruiz, Cheng Zhang, Dawen Liang, and Thang Bui, editors, *Proceedings of The 1st Symposium on Advances in Approximate Bayesian Inference*, volume 96 of *Proceedings of Machine Learning Research*, pages 32–53. PMLR, 2018. URL http://arxiv.org/abs/1805.09294.

[143] Radford M. Neal. Computing likelihood functions for high-energy physics experiments when distributions are defined by simulators with nuisance parameters. In *PHYSTAT LHC Workshop on Statistical Issues for LHC Physics, PHYSTAT 2007 - Proceedings*, pages 111–118, 2008. URL http://cds.cern.ch/record/1099977/files/p111.pdf.

[144] Yanan Fan, David J. Nott, and Scott A. Sisson. Approximate Bayesian computation via regression density estimation. *Stat*, 2 (1):34–48, dec 2013. ISSN 20491573. doi: 10.1002/sta4.15.

[145] Kyle Cranmer, Juan Pavez, and Gilles Louppe. Approximating Likelihood Ratios with Calibrated Discriminative Classifiers. *arXiv:1506.02169*, 2015. URL http://arxiv.org/abs/1506.02169.

[146] Shakir Mohamed and Balaji Lakshminarayanan. Learning in Implicit Generative Models. *arXiv:1610.03483*, oct 2016. URL http://arxiv.org/abs/1610.03483.

[147] Owen Thomas, Ritabrata Dutta, Jukka Corander, Samuel Kaski, and Michael U. Gutmann. Likelihood-free inference by ratio estimation. *arXiv:1611.10242*, nov 2016. URL http://arxiv.org/abs/1611.10242.

[148] Benjamin Miller, Alex Cole, Patrick Forré, Gilles Louppe, and Christoph Weniger. Truncated marginal neural ratio estimation. *Advances in Neural Information Processing Systems*, 34, 2021.

[149] Tuan Anh Le, Atilim Giineş Baydin, Robert Zinkov, and Frank Wood. Using synthetic data to train neural networks is model-based reasoning. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 3514–3521. IEEE, 2017.

[150] George Papamakarios and Iain Murray. Fast e-free inference of simulation models with Bayesian conditional density estimation. In *Advances in Neural Information Processing Systems*, pages 1036–1044, 2016.

[151] Jan Matthis Lueckmann, Pedro J. Gonçalves, Giacomo Bassetto, Kaan Öcal, Marcel Nonnenmacher, and Jakob H. Macke. Flexible statistical inference for mechanistic models of neural dynamics. *Advances in Neural Information Processing Systems*, 2017-December:1290–1300, nov 2017. ISSN 10495258.

[152] David Greenberg, Marcel Nonnenmacher, and Jakob Macke. Automatic posterior transformation for likelihood-free inference. In *International Conference on Machine Learning*, pages 2404–2414. PMLR, 2019.

[153] Pritam Ranjan, Derek Bingham, and George Michailidis. Sequential experiment design for contour estimation from complex computer codes. *Technometrics*, 50(4):527–541, 2008. ISSN 00401706. doi: 10.1198/004017008000000541.

[154] Julien Bect, David Ginsbourger, Ling Li, Victor Picheny, and Emmanuel Vazquez. Sequential design of computer experiments for the estimation of a probability of failure. *Statistics and Computing*, 22(3):773–793, may 2012. ISSN 09603174. doi: 10.1007/s11222-011-9241-4. URL https://doi.org/10.1007/s11222-011-9241-4.

[155] Edward Meeds and Max Welling. GPS-ABC: Gaussian process surrogate approximate Bayesian computation. In *Uncertainty in Artificial Intelligence - Proceedings of the 30th Conference, UAI 2014*, UAI'14, pages 593–602, Arlington, Virginia, United States, 2014. AUAI Press. ISBN 9780974903910. URL http://dl.acm.org/citation.cfm?id=3020751.3020813.

[156] Michael U. Gutmann and Jukka Corander. Bayesian optimization for likelihood-free inference of simulator-based statistical models. *Journal of Machine Learning Research*, 17(1):1–47, jan 2016. ISSN 15337928. URL http://dl.acm.org/citation.cfm?id=2946645.3007078.

[157] Edward Meeds and Max Welling. Optimization Monte Carlo: Efficient and embarrassingly parallel likelihood-free inference. *Advances in Neural Information Processing Systems*, 2015-January:2080–2088, jun 2015. ISSN 10495258.

[158] Marko Järvenpää, Michael U Gutmann, Arijus Pleska, Aki Vehtari, and Pekka Marttinen. Efficient acquisition rules for model-based approximate bayesian computation. *Bayesian Analysis*, 14(2):595–622, 2019.

[159] Matthew M. Graham and Amos J. Storkey. Asymptotically exact inference in differentiable generative models. *Electronic Journal of Statistics*, 11(2):5105–5164, 2017. ISSN 19357524. doi: 10.1214/17-EJS1340SI.

[160] Johann Brehmer, Gilles Louppe, Juan Pavez, and Kyle Cranmer. Mining gold from implicit models to improve likelihood-free inference. *Proceedings of the National Academy of Sciences of the United States of America*, 117(10):5242–5249, mar 2020. ISSN 10916490. doi: 10.1073/pnas.1915980117. URL http://arxiv.org/abs/1805.12244{%}0Ahttp://dx.doi.org/10.1073/pnas.1915980117http://www.ncbi.nlm.nih.gov/pubmed/32079725.

[161] Markus Stoye, Johann Brehmer, Gilles Louppe, Juan Pavez, and Kyle Cranmer. Likelihood-free inference with an improved cross-entropy estimator. *arXiv:1808.00973*, 2018. URL http://arxiv.org/abs/1808.00973.

[162] Atılım Güneş Baydin, Lukas Heinrich, Wahid Bhimji, Lei Shao, Saeid Naderiparizi, Andreas Munk, Jialin Liu, Bradley Gram-Hansen, Gilles Louppe, Lawrence Meadows, Philip Torr, Victor Lee, Prabhat, Kyle Cranmer, and Frank Wood. Efficient probabilistic inference in the quest for physics beyond the standard model. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, 2019.

[163] Hans Kersting, Nicholas Krämer, Martin Schiegg, Christian Daniel, Michael Tiemann, and Philipp Hennig. Differentiable likelihoods for fast inversion of'likelihood-free'dynamical systems. In *International Conference on Machine Learning*, pages 5198–5208. PMLR, 2020.

[164] Atilim Gunes Baydin, Lei Shao, W. Bhimji, L. Heinrich, Lawrence Meadows, Jialin Liu, Andreas Munk, Saeid Naderiparizi, Bradley Gram-Hansen, Gilles Louppe, Mingfei Ma, X. Zhao, P. Torr, V. Lee, K. Cranmer, Prabhat, and Frank D. Wood. Etalumis: bringing probabilistic programming to scientific simulators at scale. *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2019.

[165] Johann Brehmer, Kyle Cranmer, Gilles Louppe, and Juan Pavez. Constraining Effective Field Theories with Machine Learning. *Phys. Rev. Lett.*, 121(11):111801, 2018. doi: 10.1103/PhysRevLett.121.111801.

[166] Johann Brehmer, Kyle Cranmer, Gilles Louppe, and Juan Pavez. A Guide to Constraining Effective Field Theories with Machine Learning. *Phys. Rev. D*, 98(5):052004, 2018. doi: 10.1103/PhysRevD.98.052004.

[167] Johann Brehmer, Felix Kling, Irina Espejo, and Kyle Cranmer. MadMiner: Machine learning-based inference for particle physics. *Comput. Softw. Big Sci.*, 4(1):3, 2020. doi: 10.1007/s41781-020-0035-2.

[168] Arnaud Delaunoy, Antoine Wehenkel, Tanja Hinderer, Samaya Nissanke, Christoph Weniger, Andrew R. Williamson, and Gilles Louppe. Lightning-Fast Gravitational Wave Parameter Inference through Neural Amortization. 10 2020.

[169] Maximilian Dax, Stephen R. Green, Jonathan Gair, Jakob H. Macke, Alessandra Buonanno, and Bernhard Schölkopf. Real-time gravitational-wave science with neural posterior estimation. 6 2021.

[170] Joeri Hermans, Nilanjan Banik, Christoph Weniger, Gianfranco Bertone, and Gilles Louppe. Towards constraining warm dark matter with stellar streams through neural simulation-based inference. *Mon. Not. Roy. Astron. Soc.*, 507(2):1999–2011, 2021. doi: 10.1093/mnras/stab2181.

[171] Johann Brehmer, Siddharth Mishra-Sharma, Joeri Hermans, Gilles Louppe, and Kyle Cranmer. Mining for Dark Matter Substructure: Inferring subhalo population properties from strong lenses with machine learning. *Astrophys. J.*, 886(1):49, 2019. doi: 10.3847/1538-4357/ab4c41.

[172] Justin Alsing, Benjamin Wandelt, and Stephen Feeney. Massive optimal data compression and density estimation for scalable, likelihood-free inference in cosmology. *Mon. Not. Roy. Astron. Soc.*, 477(3):2874–2885, 2018. doi: 10.1093/mnras/sty819.

[173] Oliver Ratmann, Ole Jørgensen, Trevor Hinkley, Michael Stumpf, Sylvia Richardson, and Carsten Wiuf. Using likelihood-free inference to compare evolutionary dynamics of the protein networks of h. pylori and p. falciparum. *PLoS Computational Biology*, 3(11):e230, 2007.

[174] Grace Avecilla, Julie Chuong, Fangfei Li, Gavin J Sherlock, David Gresham, and Yoav Ram. Simulation-based inference of evolutionary parameters from adaptation dynamics using neural networks. *bioRxiv*, 2021.

[175] Nirit Sukenik, Oleg Vinogradov, Eyal Weinreb, Menahem Segal, Anna Levina, and Elisha Moses. Neuronal circuits overcome imbalance in excitation and inhibition by adjusting connection numbers. *Proceedings of the National Academy of Sciences*, 118(12), 2021.

[176] Sean R Bittner, Agostina Palmigiano, Alex T Piet, Chunyu A Duan, Carlos D Brody, Kenneth D Miller, and John P Cunningham. Interrogating theoretical models of neural computation with emergent property inference. *bioRxiv*, page 837567, 2021.

[177] Jonathan Oesterle, Christian Behrens, Cornelius Schröder, Thoralf Hermann, Thomas Euler, Katrin Franke, Robert G Smith, Guenther Zeck, and Philipp Berens. Bayesian inference for biophysical neuron models enables stimulus optimization for retinal neuroprosthetics. *Elife*, 9:e54997, 2020.

[178] Takeshi Yoshimatsu, Philipp Bartel, Cornelius Schröder, Filip K Janiak, Francois St-Pierre, Philipp Berens, and Tom Baden. Ancestral circuits for vertebrate colour vision emerge at the first retinal synapse. *bioRxiv*, pages 2020–10, 2021.

[179] Fabio Ramos, Rafael Carvalhaes Possas, and Dieter Fox. Bayessim: adaptive domain randomization via probabilistic inference for robotics simulators. *arXiv preprint arXiv:1906.01728*, 2019.

[180] Fabio Muratore, Theo Gruner, Florian Wiese, Boris Belousov, Michael Gienger, and Jan Peters. Neural posterior domain randomization. In *5th Annual Conference on Robot Learning*, 2021.

[181] Norman Marlier, Olivier Brüls, and Gilles Louppe. Simulation-based bayesian inference for multi-fingered robotic grasping. *arXiv preprint arXiv:2109.14275*, 2021.

[182] Johann Brehmer and Kyle Cranmer. Simulation-based inference methods for particle physics. 10 2020.

[183] Jiexiong Tang, Chenwei Deng, and Guangbin Huang. Extreme learning machine for multilayer perceptron. *IEEE Transactions on Neural Networks and Learning Systems*, 27:809–821, 2016.

[184] Glen Cowan, Kyle Cranmer, Eilam Gross, and Ofer Vitells. Asymptotic formulae for likelihood-based tests of new physics. *Eur. Phys. J. C*, 71:1554, 2011. doi: 10.1140/epjc/s10052-011-1554-0. [Erratum: Eur.Phys.J.C 73, 2501 (2013)].

[185] Johann Brehmer, Sally Dawson, Samuel Homiller, Felix Kling, and Tilman Plehn. Benchmarking simplified template cross sections in WH production. *JHEP*, 11:034, 2019. doi: 10.1007/JHEP11(2019)034.

[186] Siyu Chen, Alfredo Glioti, Giuliano Panico, and Andrea Wulzer. Parametrized classifiers for optimal EFT sensitivity. *JHEP*, 05:247, 2021. doi: 10.1007/JHEP05(2021)247.

[187] Rahool Kumar Barman, Dorival Gonçalves, and Felix Kling. Machine Learning the Higgs-Top CP Phase. 10 2021.

[188] Henning Bahl and Simon Brass. Constraining CP-violation in the Higgs-top-quark interaction using machine-learning-based inference. 10 2021.

[189] K. Cranmer, J. Brehmer, and Gilles Louppe. "active sciencing" with reusable workflows. https://github.com/cranmer/active_sciencing, 2017.

[190] Gilles Louppe. Teaching machines to discover particles. https://github.com/glouppe/talk-teaching-machines-to-discover-particles, 2017.

[191] J. Pearl. Causality: Models, reasoning and inference. 2000.

[192] J. Pearl. Causal inference in statistics: An overview. *Statistics Surveys*, 3:96–146, 2009.

[193] Susan Gruber and M. J. van der Laan. An application of collaborative targeted maximum likelihood estimation in causal inference and genomics. *The International Journal of Biostatistics*, 6, 2010.

[194] M. Prosperi, Yi Guo, M. Sperrin, J. Koopman, Jae Min, Xing He, S. Rich, Mo Wang, I. Buchan, and J. Bian. Causal inference and counterfactual prediction in machine learning for actionable healthcare. *Nature Machine Intelligence*, 2:369–375, 2020.

[195] S. Athey. The impact of machine learning on economics. 2018.

[196] Panos Toulis and D. Parkes. Long-term causal effects via behavioral game theory. In *NIPS*, 2016.

[197] J. Runge, S. Bathiany, E. Bollt, G. Camps-Valls, D. Coumou, E. Deyle, C. Glymour, M. Kretschmer, M. Mahecha, J. Muñoz-Marí, E. V. van Nes, Jonas Peters, Rick Quax, M. Reichstein, M. Scheffer, B. Schölkopf, P. Spirtes, G. Sugihara, Jie Sun, Kun Zhang, and J. Zscheischler. Inferring causation from time series in earth system sciences. *Nature Communications*, 10, 2019.

[198] B. Scholkopf, Francesco Locatello, Stefan Bauer, N. Ke, Nal Kalchbrenner, Anirudh Goyal, and Yoshua Bengio. Toward causal representation learning. *Proceedings of the IEEE*, 109:612–634, 2021.

[199] J. Pearl and D. Mackenzie. The book of why: The new science of cause and effect. 2018.

[200] Maxime Peyrard and Robert West. A ladder of causal distances. *ArXiv*, abs/2005.02480, 2020.

[201] Moritz Hardt and B. Recht. Patterns, predictions, and actions: A story about machine learning. *ArXiv*, abs/2102.05242, 2021.

[202] E. Bareinboim, Juan David Correa, D. Ibeling, and Thomas F. Icard. On pearl's hierarchy and the foundations of causal inference. 2021.

[203] S. Morgan and Christopher Winship. Counterfactuals and the potential outcome model. 2014.

[204] Jonas Peters, D. Janzing, and B. Schölkopf. Elements of causal inference: Foundations and learning algorithms. 2017.

[205] D. Rubin. Estimating causal effects of treatments in randomized and nonrandomized studies. *Journal of Educational Psychology*, 66:688–701, 1974.

[206] D. Rubin. Causal inference using potential outcomes. *Journal of the American Statistical Association*, 100:322 – 331, 2005.

[207] Daniel Eaton and K. Murphy. Exact bayesian structure learning from uncertain interventions. In *AISTATS*, 2007.

[208] Liuyi Yao, Zhixuan Chu, Sheng Li, Y. Li, Jing Gao, and A. Zhang. A survey on causal inference. *ArXiv*, abs/2002.02770, 2020.

[209] B. Scholkopf. Causality for machine learning. 2019.

[210] Ruocheng Guo, Lu Cheng, Jundong Li, P. R. Hahn, and Huan Liu. A survey of learning causality with data. *ACM Computing Surveys (CSUR)*, 53:1 – 37, 2020.

[211] B. Herd and S. Miles. Detecting causal relationships in simulation models using intervention-based counterfactual analysis. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10:1 – 25, 2019.

[212] Rick L. Stevens, Valerie E. Taylor, Jeffrey A. Nichols, Arthur B. Maccabe, Katherine A. Yelick, and David Brown. Ai for science. 2020. URL https://www.anl.gov/ai-for-science-report.

[213] Xinpeng Shen, Sisi Ma, P. Vemuri, György J. Simon, Michael W. Paul Ronald Clifford R. Andrew J. William John Weiner Aisen Petersen Jack Saykin Jagust Trojanowk, M. Weiner, P. Aisen, R. Petersen, C. Jack, A. Saykin, W. Jagust, J. Trojanowki, A. Toga, L. Beckett, R. Green, John C. Morris, L. Shaw, Z. Khachaturian, G. Sorensen, M. Carrillo, L. Kuller, M. Raichle, Steven M. Paul, P. Davies, H. Fillit, F. Hefti, D. Holtzman, M. M. Mesulam, W. Potter, P. Snyder, Adam Schwartz, T. Montine, Ronald G. Thomas, M. Donohue, Sarah Walter, Devon Gessert, T. Sather, G. Jiminez, Archana B. Balasubramanian, J. Mason, Iris Sim, D. Harvey, M. Bernstein, N. Fox, P. Thompson, N. Schuff, C. DeCarli, B. Borowski, J. Gunter, M. Senjem, David T. Jones, K. Kantarci, C. Ward, R. Koeppe, N. Foster, E. Reiman, K. Chen, C. Mathis, S. Landau, N. Cairns, Erin E. Franklin, L. Taylor-Reinwald, V. Lee, M. Korecka, Michał J. Figurski, K. Crawford, S. Neu, T. Foroud, S. Potkin, K. Faber, Sungeun Kim, K. Nho, L. Thal, N. Buckholtz, M. Albert, Richard Frank, John Hsiao, J. Kaye, J. Quinn, L. Silbert, Betty Lind, Raina Carter, Sara Dolen, L. Schneider, S. Pawluczyk, Mauricio Beccera, Liberty Teodoro, B. Spann, J. Brewer, Helen Vanderswag, A. Fleisher, J. Heidebrink, Joanne L. Lord, S. Mason, Colleen S. Albers, D. Knopman, Kris A. Johnson, R. Doody, J. Villanueva-Meyer, V. Pavlik, Victoria Shibley, M. Chowdhury, S. Rountree, Mimi Dang, Y. Stern, L. Honig, K. Bell, B. Ances, Maria Carroll, Mary L. Creech, M. Mintun, Stacy Schneider, A. Oliver, D. Marson, D. Geldmacher, M. N. Love, Randall Griffith, David Clark, J. Brockington, E. Roberson, Hillel Grossman, E. Mitsis, R. Shah, L. deToledo-Morrell, R. Duara, M. Greig-Custo, W. Barker, C. Onyike, D. D'Agostino, S. Kielb, M. Sadowski, Mohammed O. Sheikh, Anaztasia Ulysse, Mrunalini Gaikwad, P. Doraiswamy, J. Petrella, S. Borges-Neto, T. Wong, Edward Coleman, S. Arnold, J. Karlawish, D. Wolk, C. Clark, Charles D. Smith, G. Jicha, Peter Hardy, P. Sinha, Elizabeth Oates, G. Conrad, O. Lopez, MaryAnn Oakley, D. M. Simpson, A. Porsteinsson, Bonnie S. Goldstein, Kim Martin, Kelly M. Makino, M. Ismail, Connie Brand, A. Preda, D. Nguyen, K. Womack, D. Mathews, M. Quiceno, A. Levey, J. Lah, J. Cellar, J. Burns, R. Swerdlow, W. Brooks, L. Apostolova, K. Tingus, E. Woo, D. Silverman, P. Lu, G. Bartzokis, N. Graff-Radford, F. Parfitt, Kim Poki-Walker, M. Farlow, A. Hake, B. Matthews, J. Brosch, S. Herring, C. V. van Dyck, R. Carson, M. Macavoy, P. Varma, H. Chertkow, H. Bergman, Chris Hosein, S. Black, B. Stefanovic, Curtis Caldwell, G. Hsiung, B. Mudge, V. Sossi, H. Feldman, M. Assaly, E. Finger, S. Pasternack, Irina Rachisky, J. Rogers, Dick Trost, A. Kertesz, C. Bernick, D. Munic, E. Rogalski, Kristine Lipowski, S. Weintraub, B. Bonakdarpour, D. Kerwin, Chuang-Kuo Wu, N. Johnson, C. Sadowsky, Teresa Villena, R. Turner, Kathleen Johnson, Brigid Reynolds, R. Sperling, Keith A. Johnson, G. Marshall, J. Yesavage, Joy L. Taylor, B. Lane, A. Rosen, J. Tinklenberg, M. Sabbagh, C. Belden, S. Jacobson, Sherye A. Sirrel, N. Kowall, R. Killiany, A. Budson, A. Norbash, P. L. Johnson, T. Obisesan, S. Wolday, J. Allard, A. Lerner, P. Ogrocki, C. Tatsuoka, Parianne Fatica, E. Fletcher, P. Maillard,

J. Olichney, O. Carmichael, S. Kittur, M. Borrie, T.-Y. Lee, R. Bartha, Sterling C. Johnson, S. Asthana, C. Carlsson, P. Tariot, Anna D. Burke, A. Milliken, Nadira Trncic, S. Reeder, V. Bates, H. Capote, M. Rainka, D. Scharre, M. Kataki, B. Kelly, E. Zimmerman, Dzintra F Celmins, Alice D. Brown, G. Pearlson, K. Blank, K. Anderson, L. Flashman, M. Seltzer, M. Hynes, R. Santulli, K. Sink, Leslie Gordineer, J. Williamson, P. Garg, Franklin S. Watkins, B. Ott, G. Tremont, L. Daiello, S. Salloway, P. Malloy, S. Correia, H. Rosen, B. Miller, D. Perry, J. Mintzer, K. Spicer, D. Bachman, N. Pomara, Raymundo T. Hernando, Antero Sarrael, S. Schultz, Karen E Smith, Hristina K Koleva, Ki Won Nam, Hyungsub Shim, N. Relkin, Gloria Chaing, Michael P. Lin, L. Ravdin, Amanda G. Smith, Balebail Ashok Raj, and Kristin Fargher. Challenges and opportunities with causal discovery algorithms: Application to alzheimer's pathophysiology. *Scientific Reports*, 10, 2020.

[214] C. Glymour, Kun Zhang, and P. Spirtes. Review of causal discovery methods based on graphical models. *Frontiers in Genetics*, 10, 2019.

[215] J. Pearl. Causal diagrams for empirical research. *Biometrika*, 82:669–688, 1995.

[216] Robert R. Tucci. Introduction to judea pearl's do-calculus. *ArXiv*, abs/1305.5506, 2013.

[217] Julius von Kügelgen, Paul K. Rubenstein, B. Schölkopf, and Adrian Weller. Optimal experimental design via bayesian optimization: active causal structure learning for gaussian process networks. *ArXiv*, abs/1910.03962, 2019.

[218] Alexander Lavin. Neuro-symbolic neurodegenerative disease modeling as probabilistic programmed deep kernels. *International Workshop on Health Intelligence*, 2021.

[219] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press, 2005. ISBN 026218253X.

[220] Greg Chance, A. Ghobrial, Kevin McAreavey, S. Lemaignan, T. Pipe, and K. Eder. On determinism of game engines used for simulation-based autonomous vehicle verification. *ArXiv*, abs/2104.06262, 2021.

[221] Arthur Juliani, Vincent-Pierre Berges, Esh Vckay, Yuan Gao, Hunter Henry, M. Mattar, and Danny Lange. Unity: A general platform for intelligent agents. *ArXiv*, abs/1809.02627, 2018.

[222] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, J. Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *ArXiv*, abs/1606.01540, 2016.

[223] Peter F. Schulam and S. Saria. What-if reasoning with counterfactual gaussian processes. *ArXiv*, abs/1703.10651, 2017.

[224] Kexin Yi, Chuang Gan, Yunzhu Li, P. Kohli, Jiajun Wu, A. Torralba, and J. Tenenbaum. Clevrer: Collision events for video representation and reasoning. *ArXiv*, abs/1910.01442, 2020.

[225] Yunzhu Li, A. Torralba, Animashree Anandkumar, D. Fox, and Animesh Garg. Causal discovery in physical systems from videos. *ArXiv*, abs/2007.00631, 2020.

[226] Tobias Gerstenberg, Noah D. Goodman, D. Lagnado, and J. Tenenbaum. A counterfactual simulation model of causal judgments for physical events. *Psychological review*, 2021.

[227] Daniel McDuff, Yale Song, Jiyoung Lee, Vibhav Vineet, Sai Vemprala, N. Gyde, Hadi Salman, Shuang Ma, K. Sohn, and Ashish Kapoor. Causalcity: Complex simulations with agency for causal discovery and reasoning. *ArXiv*, abs/2106.13364, 2021.

[228] H. Kitano. Nobel turing challenge: creating the engine for scientific discovery. *NPJ Systems Biology and Applications*, 7, 2021.

[229] Virginia Aglietti, Xiaoyu Lu, Andrei Paleyes, and Javier Gonz' alez. Causal bayesian optimization. In *AISTATS*, 2020.

[230] Zachary Chase Lipton. The mythos of model interpretability. *Queue*, 16:31 – 57, 2018.

[231] Joshua R. Loftus, Chris Russell, Matt J. Kusner, and Ricardo Silva. Causal reasoning for algorithmic fairness. *ArXiv*, abs/1805.05859, 2018.

[232] L. Oneto and Silvia Chiappa. Fairness in machine learning. *ArXiv*, abs/2012.15816, 2020.

[233] Sina Fazelpour and Zachary Chase Lipton. Algorithmic fairness from a non-ideal perspective. *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, 2020.

[234] J. Holland and John H. Miller. Artificial adaptive agents in economic theory. *The American Economic Review*, 81:365–371, 1991.

[235] E. Bonabeau. Agent-based modeling: Methods and techniques for simulating human systems. *Proceedings of the National Academy of Sciences of the United States of America*, 99:7280 – 7287, 2002.

[236] J. Farmer and D. Foley. The economy needs agent-based modelling. *Nature*, 460:685–686, 2009.

[237] Míriam R. García, J. A. Vazquez, Isabel G. Teixeira, and A. Alonso. Stochastic individual-based modeling of bacterial growth and division using flow cytometry. *Frontiers in Microbiology*, 8, 2018.

[238] K. Mainzer. Thinking in complexity - the computational dynamics of matter, mind, and mankind, 5th edition. In *Springer complexity*, 2007.

[239] M. Batty. Generative social science: Studies in agent-based computational modeling. 2008.

[240] P. Borrill and L. Tesfatsion. Agent-based modeling: The right mathematics for the social sciences? 2011.

[241] R. Sutton and A. Barto. Reinforcement learning: An introduction. *MIT*, 2018.

[242] A. Dafoe, Edward Hughes, Yoram Bachrach, Tantum Collins, Kevin R. McKee, Joel Z. Leibo, K. Larson, and T. Graepel. Open problems in cooperative ai. *ArXiv*, abs/2012.08630, 2020.

[243] W. Arthur. Designing economic agents that act like human agents: A behavioral approach to bounded rationality. *The American Economic Review*, 81:353–359, 1991.

[244] Pablo Hernandez-Leal, M. Kaisers, T. Baarslag, and E. M. D. Cote. A survey of learning in multiagent environments: Dealing with non-stationarity. *ArXiv*, abs/1707.09183, 2017.

[245] D. Silver, Aja Huang, Chris J. Maddison, A. Guez, L. Sifre, G. V. D. Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, S. Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–489, 2016.

[246] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, K. Simonyan, L. Sifre, Simon Schmitt, A. Guez, Edward Lockhart, D. Hassabis, T. Graepel, T. Lillicrap, and D. Silver. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588 7839:604–609, 2020.

[247] Oriol Vinyals, I. Babuschkin, Wojciech Czarnecki, Michaël Mathieu, Andrew Dudzik, J. Chung, David H. Choi, Richard Powell, Timo Ewalds, P. Georgiev, Junhyuk Oh, Dan Horgan, M. Kroiss, Ivo Danihelka, Aja Huang, L. Sifre, Trevor Cai, J. Agapiou, Max Jaderberg, A. Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, D. Budden, Yury Sulsky, James Molloy, T. Paine, Caglar Gulcehre, Ziyu Wang, T. Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, T. Lillicrap, K. Kavukcuoglu, D. Hassabis, C. Apps, and D. Silver. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, pages 1–5, 2019.

[248] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemyslaw Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Christopher Hesse, R. Józefowicz, Scott Gray, Catherine Olsson, Jakub W. Pachocki, Michael Petrov, Henrique Pondé de Oliveira Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, S. Sidor, Ilya Sutskever, Jie Tang, F. Wolski, and Susan Zhang. Dota 2 with large scale deep reinforcement learning. *ArXiv*, abs/1912.06680, 2019.

[249] Noam Brown, Adam Lerer, S. Gross, and T. Sandholm. Deep counterfactual regret minimization. In *ICML*, 2019.

[250] J. Neumann and O. Morgenstern. Theory of games and economic behavior. *The Journal of Philosophy*, 42:550, 1945.

[251] N. Bard, Jakob N. Foerster, A. P. S. Chandar, Neil Burch, Marc Lanctot, H. F. Song, Emilio Parisotto, Vincent Dumoulin, Subhodeep Moitra, Edward Hughes, Iain Dunning, Shibl Mourad, H. Larochelle, Marc G. Bellemare, and Michael H. Bowling. The hanabi challenge: A new frontier for ai research. *Artif. Intell.*, 280:103216, 2020.

[252] Micah Carroll, Rohin Shah, Mark K. Ho, T. Griffiths, S. Seshia, P. Abbeel, and A. Dragan. On the utility of learning about humans for human-ai coordination. In *NeurIPS*, 2019.

[253] Max Jaderberg, Wojciech Czarnecki, Iain Dunning, Luke Marris, Guy Lever, A. Castañeda, Charlie Beattie, Neil C. Rabinowitz, Ari S. Morcos, Avraham Ruderman, Nicolas Sonnerat, Tim Green, Louise Deason, Joel Z. Leibo, D. Silver, D. Hassabis, K. Kavukcuoglu, and T. Graepel. Human-level performance in 3d multiplayer games with population-based reinforcement learning. *Science*, 364:859 – 865, 2019.

[254] Bowen Baker, I. Kanitscheider, Todor Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. Emergent tool use from multi-agent autocurricula. *ArXiv*, abs/1909.07528, 2020.

[255] Sven Gronauer and K. Diepold. Multi-agent deep reinforcement learning: a survey. *Artificial Intelligence Review*, pages 1–49, 2021.

[256] Jean Inhelder Brbel Piaget and Margaret Cook. The origins of intelligence in children. 1971.

[257] Alison Gopnik, Andrew N. Meltzoff, and Patricia K. Kuhl. The scientist in the crib : minds, brains, and how children learn. 1999.

[258] Andrew G. Barto. Intrinsic motivation and reinforcement learning. In *Intrinsically Motivated Learning in Natural and Artificial Systems*, 2013.

[259] Arthur Aubret, Laëtitia Matignon, and Salima Hassas. A survey on intrinsic motivation in reinforcement learning. *ArXiv*, abs/1908.06976, 2019.

[260] Rohin Shah, Cody Wild, Steven H. Wang, Neel Alex, Brandon Houghton, William H. Guss, Sharada Prasanna Mohanty, Anssi Kanervisto, Stephanie Milani, Nicholay Topin, P. Abbeel, Stuart J. Russell, and Anca D. Dragan. The minerl basalt competition on learning from human feedback. *ArXiv*, abs/2107.01969, 2021.

[261] S. Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *FSR*, 2017.

[262] M. Balmer, K. Meister, M. Rieser, K. Nagel, and K. Axhausen. Agent-based simulation of travel demand: Structure and computational performance of matsim-t. 2008.

[263] A. Crooks and Sarah C. Wise. Gis and agent-based models for humanitarian assistance. *Comput. Environ. Urban Syst.*, 41: 100–111, 2013.

[264] T. Schoenharl, G. Madey, Gábor Szabó, and A. Barabasi. Wiper: A multi-agent system for emergency response. 2006.

[265] Nicolás Garrido and L. Mittone. An agent based model for studying optimal tax collection policy using experimental data: The cases of chile and italy. *Journal of Socio-economics*, 42:24–30, 2013.

[266] C. Angione, E. Silverman, and E. Yaneske. Using machine learning to emulate agent-based simulations. *ArXiv*, abs/2005.02077, 2020.

[267] Jiaxin Zhang. Modern monte carlo methods for efficient uncertainty quantification and propagation: A survey. *Wiley Interdisciplinary Reviews: Computational Statistics*, page e1539.

[268] Pokemon yellow screenshot. URL https://www.pokemon.com/us/pokemon-video-games/pokemon-yellow-special-pikachu-edition/.

[269] Alessandro Moro. Understanding the dynamics of violent political revolutions in an agent-based framework. *PLoS ONE*, 11, 2016.

[270] L. Overbey, B. Mitchell, Samuel Yaryan, and K. McCullough. A large scale, high resolution agent-based insurgency model. 2013.

[271] J. Bae and Il-Chul Moon. Ldef formalism for agent-based model development. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 46:793–808, 2016.

[272] Joshua M. Epstein. Modeling civil violence: an agent-based computational approach. *Proceedings of the National Academy of Sciences of the United States of America*, 99 Suppl 3:7243–50, 2002.

[273] Philip Paquette, Yuchen Lu, Steven Bocco, Max O. Smith, Satya Ortiz-Gagné, Jonathan K. Kummerfeld, Satinder P. Singh, Joelle Pineau, and Aaron C. Courville. No press diplomacy: Modeling multi-agent gameplay. *ArXiv*, abs/1909.02128, 2019.

[274] C. Kopp, K. Korb, and B. Mills. Information-theoretic models of deception: Modelling cooperation and diffusion in populations exposed to "fake news". *PLoS ONE*, 13, 2018.

[275] Mohammad Soheilypour and Mohammad R. Kaazempur Mofrad. Agent-based modeling in molecular systems biology. *BioEssays*, 40, 2018.

[276] Mohammad Azimi, Evgeny Bulat, Karsten Weis, and Mohammad R. K. Mofrad. An agent-based model for mrna export through the nuclear pore complex. *Molecular Biology of the Cell*, 25:3643 – 3653, 2014.

[277] Zachary B Katz, Brian P. English, Timothée Lionnet, Young J. Yoon, Nilah Monnier, Ben Ovryn, Mark Bathe, and Robert H. Singer. Mapping translation 'hot-spots' in live cells by tracking single molecules of mrna and ribosomes. *eLife*, 5, 2016.

[278] M. Soheilypour and M. Mofrad. Regulation of rna-binding proteins affinity to export receptors enables the nuclear basket proteins to distinguish and retain aberrant mrnas. *Scientific Reports*, 6, 2016.

[279] Gaelle Letort, Arnau Montagud, Gautier Stoll, Randy W. Heiland, Emmanuel Barillot, Paul Macklin, Andrei Yu. Zinovyev, and Laurence Calzone. Physiboss: a multi-scale agent-based modelling framework integrating physical dimension and cell signalling. *Bioinformatics*, 35:1188 – 1196, 2019.

[280] Rory Greig and Jordi Arranz. Generating agent-based models from scratch with genetic programming. 2021.

[281] Allegra A Beal Cohen, Rachata Muneepeerakul, and Gregory A. Kiker. Intra-group decision-making in agent-based models. *Scientific Reports*, 11, 2021.

[282] Ross A. Hammond. Considerations and best practices in agent-based modeling to inform policy. 2015.

[283] Natasha Jaques, Angeliki Lazaridou, Edward Hughes, Çaglar Gülçehre, Pedro A. Ortega, DJ Strouse, Joel Z. Leibo, and Nando de Freitas. Social influence as intrinsic motivation for multi-agent deep reinforcement learning. In *ICML*, 2019.

[284] Jan-Willem van de Meent, Brooks Paige, Hongseok Yang, and Frank Wood. An introduction to probabilistic programming. *arXiv preprint arXiv:1809.10756*, 2018.

[285] A. Pfeffer, Brian E. Ruttenberg, William Kretschmer, and Alison O'Connor. Structured factored inference for probabilistic programming. In *AISTATS*, 2018.

[286] A. Gordon, T. Henzinger, A. Nori, and S. Rajamani. Probabilistic programming. *Future of Software Engineering Proceedings*, 2014.

[287] Noah D. Goodman, Vikash K. Mansinghka, D. M. Roy, Keith Bonawitz, and J. Tenenbaum. Church: a language for generative models. In *UAI*, 2008.

[288] A. Pfeffer. Figaro : An object-oriented probabilistic programming language. 2009.

[289] Vikash K. Mansinghka, Daniel Selsam, and Yura N. Perov. Venture: a higher-order probabilistic programming platform with programmable inference. *ArXiv*, abs/1404.0099, 2014.

[290] F. Wood, Jan-Willem van de Meent, and Vikash K. Mansinghka. A new approach to probabilistic programming inference. In *AISTATS*, 2014.

[291] Noah D. Goodman and Andreas Stuhlmüller. The Design and Implementation of Probabilistic Programming Languages. http://dippl.org. Accessed: 2014-08-27.

[292] D. Lunn, D. Spieǵelhalter, A. Thomas, and N. Best. The bugs project: Evolution, critique and future directions. *Statistics in medicine*, 28 25:3049–67, 2009.

[293] John Winn and Tom Minka. Probabilistic programming with infer.net. 2009. URL https://www.microsoft.com/en-us/research/publication/probabilistic-programming-infer-net/.

[294] Brian Milch, Bhaskara Marthi, S. Russell, D. Sontag, D. L. Ong, and A. Kolobov. Blog: Probabilistic models with unknown objects. In *IJCAI*, 2005.

[295] Dustin Tran, Alp Kucukelbir, Adji B. Dieng, M. Rudolph, Dawen Liang, and D. Blei. Edward: A library for probabilistic modeling, inference, and criticism. *ArXiv*, abs/1610.09787, 2016.

[296] Andreas Munk, A. Scibior, Atilim Gunes Baydin, Andrew Stewart, G. Fernlund, A. Poursartip, and Frank D. Wood. Deep probabilistic surrogate networks for universal simulator approximation. *ArXiv*, abs/1910.11950, 2019.

[297] S. Weber, A. Gelman, D. Lee, M. Betancourt, Aki Vehtari, and A. Racine. Bayesian aggregation of average data: An application in drug development. 2018.

[298] David Merrell and A. Gitter. Inferring signaling pathways with probabilistic programming. *Bioinformatics*, 36:i822 – i830, 2020.

[299] L. Ouyang, Michael Henry Tessler, Daniel Ly, and Noah D. Goodman. Practical optimal experiment design with probabilistic programs. *ArXiv*, abs/1608.05046, 2016.

[300] Frank D. Wood, Andrew Warrington, Saeid Naderiparizi, Christian Weilbach, Vaden Masrani, William Harvey, A. Scibior, Boyan Beronov, and Alireza Nasseri. Planning as inference in epidemiological models. *ArXiv*, abs/2003.13221, 2020.

[301] C. S. D. Witt, Bradley Gram-Hansen, Nantas Nardelli, Andrew Gambardella, R. Zinkov, P. Dokania, N. Siddharth, Ana Belén Espinosa-González, A. Darzi, Philip H. S. Torr, and Atilim Gunes Baydin. Simulation-based inference for global health decisions. *ArXiv*, abs/2005.07062, 2020.

[302] Sylvia L. Ranjeva, J. Mihaljevic, M. Joseph, A. Giuliano, and G. Dwyer. Untangling the dynamics of persistence and colonization in microbial communities. *The ISME Journal*, 13:2998 – 3010, 2019.

[303] M. Joseph, M. Rossi, Nathan P. Mietkiewicz, Adam L. Mahood, M. Cattau, L. A. St. Denis, R. C. Nagy, V. Iglesias, J. Abatzoglou, and Jennifer K. Balch. Spatiotemporal prediction of wildfire size extremes with bayesian finite sample maxima. *Ecological Applications*, 29, 2019.

[304] G. Acciarini, Francesco Pinto, Sascha Metz, Sarah Boufelja, S. Kaczmarek, K. Merz, Jose Martinez-Heras, F. Letizia, C. Bridges, and Atilim Gunes Baydin. Spacecraft collision risk assessment with probabilistic programming. *ArXiv*, abs/2012.10260, 2020.

[305] Hong Ge, Kai Xu, and Zoubin Ghahramani. Turing: A language for flexible probabilistic inference. 2018.

[306] Valentin Dalibard, Michael Schaarschmidt, and E. Yoneki. Boat: Building auto-tuners with structured bayesian optimization. *Proceedings of the 26th International Conference on World Wide Web*, 2017.

[307] Alexander Lavin. Doubly bayesian optimization. *ArXiv*, abs/1812.04562, 2018.

[308] Tuan Anh Le, Atilim Gunes Baydin, and Frank D. Wood. Inference compilation and universal probabilistic programming. In *AISTATS*, 2017.

[309] William Harvey, Andreas Munk, Atilim Gunes Baydin, Alexander Bergholm, and Frank D. Wood. Attention for inference compilation. *ArXiv*, abs/1910.11961, 2019.

[310] Mario Lezcano Casado, Atilim Gunes Baydin, David Martínez-Rubio, Tuan Anh Le, Frank D. Wood, L. Heinrich, Gilles Louppe, K. Cranmer, Karen Ng, W. Bhimji, and Prabhat. Improvements to inference compilation for probabilistic programming in large-scale scientific simulators. *ArXiv*, abs/1712.07901, 2017.

[311] Bradley Gram-Hansen, Christian Schroeder, Philip H.S. Torr, Yee Whye Teh, Tom Rainforth, and Atılım Güneş Baydin. Hijacking malaria simulators with probabilistic programming. In *ICML Workshop on AI for Social Good, Thirty-sixth International Conference on Machine Learning (ICML 2019), Long Beach, CA, US*, 2019.

[312] Enrico Bothmann, Gurpreet Singh Chahal, Stefan Höche, Johannes Krause, Frank Krauss, Silvan Kuttimalai, Sebastian Liebschner, Davide Napoletano, Marek Schönherr, Holger Schulz, Steffen Schumann, and Frank Siegert. Event generation with sherpa 2.2. *SciPost Physics Proceedings*, 7(3), 9 2019. ISSN 2542-4653. doi: 10.21468/SciPostPhys.7.3.034. URL https://www.osti.gov/biblio/1562545.

[313] Christian Schroeder de Witt, Bradley Gram-Hansen, Nantas Nardelli, Andrew Gambardella, Rob Zinkov, Puneet Dokania, N. Siddharth, Ana Belen Espinosa-Gonzalez, Ara Darzi, Philip Torr, and Atılım Güneş Baydin. Simulation-based inference for global health decisions. In *ICML Workshop on Machine Learning for Global Health, Thirty-seventh International Conference on Machine Learning (ICML 2020)*, 2020.

[314] Frank Wood, Andrew Warrington, Saeid Naderiparizi, Christian Weilbach, Vaden Masrani, William Harvey, Adam Scibior, Boyan Beronov, John Grefenstette, Duncan Campbell, et al. Planning as inference in epidemiological models. *arXiv preprint arXiv:2003.13221*, 2020.

[315] A. Bershteyn, J. Gerardin, Daniel Bridenbecker, Christopher W. Lorton, J. Bloedow, R. S. Baker, G. Chabot-Couture, Y. Chen, Thomas Fischle, Kurt Frey, J. Gauld, H. Hu, A. Izzo, D. Klein, Dejan Lukacevic, K. McCarthy, J. Miller, A. Ouédraogo, T. Perkins, Jeffrey Steinkraus, Q. T. ten Bosch, Hung-Fu Ting, S. Titova, B. Wagner, P. Welkhoff, E. Wenger, and Christian N Wiswell. Implementation and applications of emod, an individual-based multi-disease modeling platform. *Pathogens and Disease*, 76, 2018.

[316] T. Smith, N. Maire, A. Ross, M. Penny, N. Chitnis, A. Schapira, A. Studer, B. Genton, C. Lengeler, F. Tediosi, D. de Savigny, and M. Tanner. Towards a comprehensive simulation model of malaria epidemiology and control. *Parasitology*, 135:1507 – 1516, 2008.

[317] John Bradshaw, Brooks Paige, Matt J. Kusner, Marwin H. S. Segler, and José Miguel Hernández-Lobato. Barking up the right tree: an approach to search over molecule synthesis dags. *ArXiv*, abs/2012.11522, 2020.

[318] B. Lake, Tomer D. Ullman, J. Tenenbaum, and S. Gershman. Building machines that learn and think like people. *Behavioral and Brain Sciences*, 40, 2016.

[319] D. George, Wolfgang Lehrach, Ken Kansky, M. Lázaro-Gredilla, C. Laan, B. Marthi, Xinghua Lou, Zhaoshi Meng, Y. Liu, Huayan Wang, Alexander Lavin, and D. Phoenix. A generative vision model that trains with high data efficiency and breaks text-based captchas. *Science*, 358, 2017.

[320] G. Marcus and E. Davis. Insights for ai from the human mind. *Communications of the ACM*, 64:38 – 41, 2021.

[321] Ernő Téglás, E. Vul, V. Girotto, Michel Gonzalez, J. Tenenbaum, and L. Bonatti. Pure reasoning in 12-month-old infants as probabilistic inference. *Science*, 332:1054 – 1059, 2011.

[322] Shari Liu, Tomer D. Ullman, J. Tenenbaum, and E. Spelke. Ten-month-old infants infer the value of goals from the costs of actions. *Science*, 358:1038 – 1041, 2017.

[323] Tomer D. Ullman, E. Spelke, P. Battaglia, and J. Tenenbaum. Mind games: Game engines as an architecture for intuitive physics. *Trends in Cognitive Sciences*, 21:649–665, 2017.

[324] Tomer D. Ullman and J. Tenenbaum. Bayesian models of conceptual development: Learning as building models of the world. 2020.

[325] S. Gershman, Tobias Gerstenberg, Chris L. Baker, and F. Cushman. Plans, habits, and theory of mind. *PLoS ONE*, 11, 2016.

[326] P. Battaglia, Jessica B. Hamrick, and J. Tenenbaum. Simulation as an engine of physical scene understanding. *Proceedings of the National Academy of Sciences*, 110:18327 – 18332, 2013.

[327] Jessica B. Hamrick, P. Battaglia, T. Griffiths, and J. Tenenbaum. Inferring mass in complex scenes by mental simulation. *Cognition*, 157:61–76, 2016.

[328] Chris L. Baker, J. Jara-Ettinger, R. Saxe, and J. Tenenbaum. Rational quantitative attribution of beliefs, desires and percepts in human mentalizing. *Nature Human Behaviour*, 1, 2017.

[329] Vikash K. Mansinghka, Tejas D. Kulkarni, Yura N. Perov, and J. Tenenbaum. Approximate bayesian image interpretation using generative probabilistic graphics programs. In *NIPS*, 2013.

[330] Tejas D. Kulkarni, P. Kohli, J. Tenenbaum, and Vikash K. Mansinghka. Picture: A probabilistic programming language for scene perception. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4390–4399, 2015.

[331] Zhihua Wang, Stefano Rosa, Bo Yang, Sen Wang, A. Trigoni, and A. Markham. 3d-physnet: Learning the intuitive physics of non-rigid object deformations. In *IJCAI*, 2018.

[332] B. Lake, R. Salakhutdinov, and J. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350:1332 – 1338, 2015.

[333] Joshua Rule, J. Tenenbaum, and S. Piantadosi. The child as hacker. *Trends in Cognitive Sciences*, 24:900–915, 2020.

[334] Kevin Ellis, Catherine Wong, Maxwell Nye, Mathias Sablé-Meyer, Luc Cary, Lucas Morales, Luke Hewitt, Armando Solar-Lezama, and J. Tenenbaum. Dreamcoder: Growing generalizable, interpretable knowledge with wake-sleep bayesian program learning. *ArXiv*, abs/2006.08381, 2020.

[335] A. Heinecke, Alexander Breuer, Sebastian Rettenberger, M. Bader, A. Gabriel, C. Pelties, A. Bode, W. Barth, Xiangke Liao, K. Vaidyanathan, M. Smelyanskiy, and P. Dubey. Petascale high order dynamic rupture earthquake simulations on heterogeneous supercomputers. *SC14: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 3–14, 2014.

[336] E. Endeve, C. Cardall, R. Budiardja, S. Beck, Alborz Bejnood, R. Toedte, A. Mezzacappa, and J. Blondin. Turbulent magnetic field amplification from spiral sasi modes: Implications for core-collapse supernovae and proto-neutron star magnetization. *The Astrophysical Journal*, 751:26, 2012.

[337] M. Raberto, S. Cincotti, S. Focardi, and M. Marchesi. Agent-based simulation of a financial market. *Physica A-statistical Mechanics and Its Applications*, 299:319–327, 2001.

[338] A. Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *ArXiv*, abs/1410.5401, 2014.

[339] A. Graves, Greg Wayne, M. Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwinska, Sergio Gomez Colmenarejo, Edward Grefenstette, Tiago Ramalho, J. Agapiou, Adrià Puigdomènech Badia, K. Hermann, Yori Zwols, Georg Ostrovski, Adam Cain, Helen King, C. Summerfield, P. Blunsom, K. Kavukcuoglu, and D. Hassabis. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538:471–476, 2016.

[340] M. Betancourt. A conceptual introduction to hamiltonian monte carlo. *arXiv: Methodology*, 2017.

[341] M. Hoffman and A. Gelman. The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo. *J. Mach. Learn. Res.*, 15:1593–1623, 2014.

[342] Christos Louizos, U. Shalit, J. Mooij, D. Sontag, R. Zemel, and M. Welling. Causal effect inference with deep latent-variable models. *ArXiv*, abs/1705.08821, 2017.

[343] C. John. Latent variable models: an introduction to factor, path, and structural analysis. 1986.

[344] Atilim Gunes Baydin, Barak A. Pearlmutter, Alexey Radul, and J. Siskind. Automatic differentiation in machine learning: a survey. *J. Mach. Learn. Res.*, 18:153:1–153:43, 2017.

[345] Edward Grefenstette, K. Hermann, Mustafa Suleyman, and P. Blunsom. Learning to transduce with unbounded memory. In *NIPS*, 2015.

[346] J. Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph P. Turian, David Warde-Farley, and Yoshua Bengio. Theano: A cpu and gpu math compiler in python. 2010.

[347] Martín Abadi, P. Barham, Jianmin Chen, Z. Chen, Andy Davis, J. Dean, M. Devin, S. Ghemawat, Geoffrey Irving, M. Isard, M. Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, D. Murray, Benoit Steiner, P. Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Y. Yu, and Xiaoqiang Zhang. Tensorflow: A system for large-scale machine learning. In *OSDI*, 2016.

[348] Adam Paszke, S. Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zach DeVito, Zeming Lin, Alban Desmaison, L. Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

[349] Roy Frostig, M. Johnson, and Chris Leary. Compiling machine learning programs via high-level tracing. 2018.

[350] Yuanming Hu, Luke Anderson, Tzu-Mao Li, Qi Sun, N. Carr, Jonathan Ragan-Kelley, and F. Durand. Difftaichi: Differentiable programming for physical simulation. *ArXiv*, abs/1910.00935, 2020.

[351] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *SIAM review*, 59(1):65–98, 2017. URL https://doi.org/10.1137/141000671.

[352] Mike Innes, Alan Edelman, Keno Fischer, Chris Rackauckas, Elliot Saba, Viral B. Shah, and Will Tebbutt. A differentiable programming system to bridge machine learning and scientific computing. *ArXiv*, abs/1907.07587, 2019.

[353] Andrzej Kolinski. Multiscale approaches to protein modeling. 2011.

[354] Sergey Ovchinnikov, Hahnbeom Park, Neha Varghese, Po-Ssu Huang, Georgios A. Pavlopoulos, David E. Kim, Hetunandan Kamisetty, Nikos C. Kyrpides, and David Baker. Protein structure determination using metagenome sequence data. *Science*, 355:294 – 298, 2017.

[355] Thomas A. Hopf, John Ingraham, Frank J. Poelwijk, Charlotta P I Schärfe, Michael Springer, Chris Sander, and Debora S. Marks. Mutation effects predicted from sequence co-variation. *Nature Biotechnology*, 35:128–135, 2017.

[356] Mohammed AlQuraishi. End-to-end differentiable learning of protein structure. *Cell systems*, 8 4:292–301.e3, 2019.

[357] Chengxin Zhang, S. M. Mortuza, Baoji He, Yanting Wang, and Yang Arthur Zhang. Template-based and free modeling of i-tasser and quark pipelines using predicted contact maps in casp12. *Proteins: Structure*, 86:136 – 151, 2018.

[358] John M Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Zídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A A Kohl, Andy Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David A. Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with alphafold. *Nature*, 596:583 – 589, 2021.

[359] Benjamin R. Jagger, Sarah E. Kochanek, Susanta Haldar, Rommie E. Amaro, and Adrian J. Mulholland. Multiscale simulation approaches to modeling drug-protein binding. *Current opinion in structural biology*, 61:213–221, 2020.

[360] Joshua A. Anderson, Jens Glaser, and Sharon C. Glotzer. Hoomd-blue: A python package for high-performance molecular dynamics and hard particle monte carlo simulations. *arXiv: Computational Physics*, 2013.

[361] Steven J. Plimpton. Fast parallel algorithms for short-range molecular dynamics. *Journal of Computational Physics*, 117: 1–19, 1993.

[362] Aidan P. Thompson, H. Metin Aktulga, Richard Berger, Dan S. Bolintineanu, W. Michael Brown, Paul S. Crozier, Pieter J. in 't Veld, Axel Kohlmeyer, Stan G. Moore, Trung Dac Nguyen, Ray Shan, Mark Stevens, Julien Tranchida, Christian R. Trott, and Steven J. Plimpton. Lammps - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales. *Computer Physics Communications*, 2021.

[363] Laurent Li, Stephan Hoyer, Ryan Pederson, Ruoxi Sun, Ekin Dogus Cubuk, Patrick F. Riley, and Kieron Burke. Kohn-sham equations as regularizer: building prior knowledge into machine-learned physics. *Physical review letters*, 126 3:036401, 2021.

[364] Marc Finzi, Max Welling, and Andrew Gordon Wilson. A practical method for constructing equivariant multilayer perceptrons for arbitrary matrix groups. *ArXiv*, abs/2104.09459, 2021.

[365] C. Daniel Freeman, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem. Brax - a differentiable physics engine for large scale rigid body simulation. *ArXiv*, abs/2106.13281, 2021.

[366] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, pages 1530–1538. PMLR, 2015.

[367] George Papamakarios, Eric T. Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *ArXiv*, abs/1912.02762, 2019.

[368] Yuanming Hu, Tzu-Mao Li, Luke Anderson, Jonathan Ragan-Kelley, and Frédo Durand. Taichi: a language for high-performance computation on spatially sparse data structures. *ACM Trans. Graph.*, 38:201:1–201:16, 2019.

[369] Chris Lattner and Vikram S. Adve. Llvm: a compilation framework for lifelong program analysis & transformation. *International Symposium on Code Generation and Optimization, 2004. CGO 2004.*, pages 75–86, 2004.

[370] Yann LeCun. Deep learning est mort. vive differentiable programming! 2018. URL https://www.facebook.com/yann.lecun/posts/10155003011462143.

[371] M. Abadi and G. Plotkin. A simple differentiable programming language. *Proceedings of the ACM on Programming Languages*, 4:1 – 28, 2020.

[372] W. Banzhaf, Bert O. Baumgaertner, G. Beslon, R. Doursat, J. Foster, B. McMullin, Vinicius Veloso de Melo, Thomas Miconi, L. Spector, S. Stepney, and Roger White. Defining and simulating open-ended novelty: requirements, guidelines, and challenges. *Theory in Biosciences*, 135:131–161, 2016.

[373] Susan Stepney. Modelling and measuring open-endedness. *Artificial Life*, 25(1):9, 2021.

[374] Karl J. Friston, Marco Lin, Christopher D. Frith, Giovanni Pezzulo, J. Allan Hobson, and Sasha Ondobaka. Active Inference, Curiosity and Insight. *Neural Computation*, 29(10):2633–2683, October 2017. ISSN 0899-7667, 1530-888X. doi: 10.1162/neco_a_00999.

[375] David Krakauer, Nils Bertschinger, Eckehard Olbrich, Jessica C. Flack, and Nihat Ay. The information theory of individuality. *Theory Biosci.*, 139(2):209–223, June 2020. ISSN 1431-7613, 1611-7530. doi: 10.1007/s12064-020-00313-7.

[376] Steen Rasmussen and P. Sibani. Two modes of evolution: Optimization and expansion. *Artificial Life*, 25:9–21, 2019.

[377] Alyssa M. Adams, Hector Zenil, Paul C. W. Davies, and Sara Imari Walker. Formal definitions of unbounded evolution and innovation reveal universal mechanisms for open-ended evolution in dynamical systems. *Scientific Reports*, 7, 2017.

[378] Joel Lehman and Kenneth O. Stanley. Novelty Search and the Problem with Objectives. In Rick Riolo, Ekaterina Vladislavleva, and Jason H. Moore, editors, *Genetic Programming Theory and Practice IX*, pages 37–56. Springer New York, New York, NY, 2011. ISBN 978-1-4614-1769-9 978-1-4614-1770-5. doi: 10.1007/978-1-4614-1770-5_3.

[379] Dharshan Kumaran, Demis Hassabis, and James L. McClelland. What learning systems do intelligent agents need? complementary learning systems theory updated. *Trends in Cognitive Sciences*, 20:512–534, 2016.

[380] Timothy J. Taylor. From artificial evolution to artificial life. 1999.

[381] Santiago Hernández-Orozco, Francisco Hernández Quiroz, and Hector Zenil. Undecidability and irreducibility conditions for open-ended evolution and emergence. *Artificial Life*, 24:56–70, 2018.

[382] Santiago Hernández-Orozco, Narsis Aftab Kiani, and Hector Zenil. Algorithmically probable mutations reproduce aspects of evolution, such as convergence rate, genetic memory and modularity. *Royal Society Open Science*, 5, 2018.

[383] Edmund Burke and Graham Kendall, editors. *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*. Springer, New York, 2005. ISBN 978-0-387-23460-1 978-0-387-28356-2.

[384] Sebastian Thrun. Lifelong Learning Algorithms. In Sebastian Thrun and Lorien Pratt, editors, *Learning to Learn*, pages 181–209. Springer US, Boston, MA, 1998. ISBN 978-1-4613-7527-2 978-1-4615-5529-2. doi: 10.1007/978-1-4615-5529-2_8.

[385] Elizabeth Bonawitz, Stephanie Denison, Alison Gopnik, and Thomas L. Griffiths. Win-Stay, Lose-Sample: A simple sequential algorithm for approximating Bayesian inference. *Cognitive Psychology*, 74:35–65, November 2014. ISSN 00100285. doi: 10.1016/j.cogpsych.2014.06.003.

[386] Philipp Hennig and Christian J. Schuler. Entropy search for information-efficient global optimization. *ArXiv*, abs/1112.1217, 2012.

[387] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic Algorithms and Applications. *arXiv:1812.05905 [cs, stat]*, December 2018.

[388] José Miguel Hernández-Lobato, Matthew W. Hoffman, and Zoubin Ghahramani. Predictive entropy search for efficient global optimization of black-box functions. In *NIPS*, 2014.

[389] Ilya M. Sobol. On the distribution of points in a cube and the approximate evaluation of integrals. *Ussr Computational Mathematics and Mathematical Physics*, 7:86–112, 1967.

[390] Christiane Lemieux. Monte carlo and quasi-monte carlo sampling. 2009.

[391] Benjamin Letham, Brian Karrer, Guilherme Ottoni, and Eytan Bakshy. Constrained bayesian optimization with noisy experiments. *ArXiv*, abs/1706.07094, 2019.

[392] Joel Lehman and Kenneth O. Stanley. Abandoning Objectives: Evolution Through the Search for Novelty Alone. *Evolutionary Computation*, 19(2):189–223, June 2011. ISSN 1063-6560, 1530-9304. doi: 10.1162/EVCO_a_00025.

[393] Erik J Peterson and Timothy D Verstynen. Curiosity eliminates the exploration-exploitation dilemma. Preprint, Bioa, June 2019.

[394] Iztok Fister, Andres Iglesias, Akemi Galvez, Javier Del Ser, Eneko Osaba, Iztok Fister, Matjaž Perc, and Mitja Slavinec. Novelty search for global optimization. *Applied Mathematics and Computation*, 347:865–881, April 2019. ISSN 00963003. doi: 10.1016/j.amc.2018.11.052.

[395] Jean-Baptiste Mouret. Novelty-Based Multiobjectivization. In Janusz Kacprzyk, Stéphane Doncieux, Nicolas Bredèche, and Jean-Baptiste Mouret, editors, *New Horizons in Evolutionary Robotics*, volume 341, pages 139–154. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. ISBN 978-3-642-18271-6 978-3-642-18272-3. doi: 10.1007/978-3-642-18272-3_10.

[396] Stephane Doncieux, Giuseppe Paolo, Alban Laflaquière, and Alexandre Coninx. Novelty Search makes Evolvability Inevitable. *arXiv:2005.06224 [cs]*, May 2020.

[397] Hideyuki Takagi. Interactive evolutionary computation: fusion of the capabilities of ec optimization and human evaluation. *Proc. IEEE*, 89:1275–1296, 2001.

[398] Brian G. Woolley and Kenneth O. Stanley. A novel human-computer collaboration: combining novelty search with interactive evolution. *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, 2014.

[399] Mathias Löwe and Sebastian Risi. Accelerating the evolution of cognitive behaviors through human-computer collaboration. *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, 2016.

[400] D. E. Berlyne. A theory of human curiosity. *British Journal of Psychology. General Section*, 45(3):180–191, 1954.

[401] Celeste Kidd and Benjamin Y. Hayden. The Psychology and Neuroscience of Curiosity. *Neuron*, 88(3):449–460, November 2015. ISSN 08966273. doi: 10.1016/j.neuron.2015.09.010.

[402] Jacqueline Gottlieb and Pierre-Yves Oudeyer. Towards a neuroscience of active sampling and curiosity. *Nat Rev Neurosci*, 19 (12):758–770, December 2018. ISSN 1471-003X, 1471-0048. doi: 10.1038/s41583-018-0078-0.

[403] Nikolay Savinov, Anton Raichuk, Raphaël Marinier, Damien Vincent, Marc Pollefeys, Timothy Lillicrap, and Sylvain Gelly. Episodic Curiosity through Reachability. *arXiv:1810.02274 [cs, stat]*, August 2019.

[404] Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei A. Efros. Large-Scale Study of Curiosity-Driven Learning. *arXiv:1808.04355 [cs, stat]*, August 2018.

[405] Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. Go-Explore: A New Approach for Hard-Exploration Problems. *arXiv:1901.10995 [cs, stat]*, May 2019.

[406] Eliza Kosoy, Jasmine Collins, David M. Chan, Sandy Huang, Deepak Pathak, Pulkit Agrawal, John Canny, Alison Gopnik, and Jessica B. Hamrick. Exploring Exploration: Comparing Children with RL Agents in Unified Environments. *arXiv:2005.02880 [cs]*, July 2020.

[407] Cédric Colas, Pierre Fournier, Olivier Sigaud, Mohamed Chetouani, and Pierre-Yves Oudeyer. CURIOUS: Intrinsically Motivated Multi-Task Multi-Goal Reinforcement Learning. *Arxiv*, 1810.06284v3:1–15, 2019.

[408] Pierre-Yves Oudeyer. Computational Theories of Curiosity-Driven Learning. *arXiv:1802.10546 [cs]*, June 2018.

[409] Timnit Gebru, Judy Hoffman, and Li Fei-Fei. Fine-Grained Recognition in the Wild: A Multi-task Domain Adaptation Approach. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 1358–1367, Venice, October 2017. IEEE. ISBN 978-1-5386-1032-9. doi: 10.1109/ICCV.2017.151.

[410] Eun Seo Jo and Timnit Gebru. Lessons from archives: Strategies for collecting sociocultural data in machine learning. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, pages 306–316, Barcelona Spain, January 2020. ACM. ISBN 978-1-4503-6936-7. doi: 10.1145/3351095.3372829.

[411] Olivier Sigaud, Hugo Caselles-Dupré, Cédric Colas, Ahmed Akakzia, Pierre-Yves Oudeyer, and Mohamed Chetouani. Towards Teachable Autonomous Agents. *arXiv:2105.11977 [cs]*, May 2021.

[412] Adrien Laversanne-Finot, Alexandre Péré, and Pierre-Yves Oudeyer. Curiosity Driven Exploration of Learned Disentangled Goal Spaces. *arXiv:1807.01521 [cs, stat]*, November 2018.

[413] Alison Gopnik and Henry M. Wellman. Reconstructing constructivism: Causal models, Bayesian learning mechanisms, and the theory theory. *Psychological Bulletin*, 138(6):1085–1108, 2012. ISSN 1939-1455, 0033-2909. doi: 10.1037/a0028044.

[414] Sumedh A. Sontakke, Arash Mehrjou, Laurent Itti, and Bernhard Schölkopf. Causal Curiosity: RL Agents Discovering Self-supervised Experiments for Causal Representation Learning. *arXiv:2010.03110 [cs]*, October 2020.

[415] Zachary C Lipton, Kamyar Azizzadenesheli, Abhishek Kumar, Lihong Li, Jianfeng Gao, and Li Deng. Combating Reinforcement Learning's Sisyphean Curse with Intrinsic Fear. *Arxiv*, 1611.01211:14, 2018.

[416] Adrien Baranes and Pierre-Yves Oudeyer. Active learning of inverse models with intrinsically motivated goal exploration in robots. *Robotics and Autonomous Systems*, 61(1):49–73, January 2013. ISSN 09218890. doi: 10.1016/j.robot.2012.05.008.

[417] Justin K. Pugh, Lisa B. Soros, and Kenneth O. Stanley. Quality Diversity: A New Frontier for Evolutionary Computation. *Front. Robot. AI*, 3, July 2016. ISSN 2296-9144. doi: 10.3389/frobt.2016.00040.

[418] Cédric Colas, Joost Huizinga, Vashisht Madhavan, and Jeff Clune. Scaling MAP-Elites to Deep Neuroevolution. *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, pages 67–75, June 2020. doi: 10.1145/3377930.3390217.

[419] Jean-Baptiste Mouret and Jeff Clune. Illuminating search spaces by mapping elites. *arXiv:1504.04909 [cs, q-bio]*, April 2015.

[420] Roby Velez and Jeff Clune. Novelty search creates robots with general skills for exploration. In *Proceedings of the 2014 Conference on Genetic and Evolutionary Computation - GECCO '14*, pages 737–744, Vancouver, BC, Canada, 2014. ACM Press. ISBN 978-1-4503-2662-9. doi: 10.1145/2576768.2598225.

[421] Antoine Cully, Jeff Clune, Danesh Tarapore, and Jean-Baptiste Mouret. Robots that can adapt like animals. *Nature*, 521 (7553):503–507, May 2015. ISSN 0028-0836, 1476-4687. doi: 10.1038/nature14422.

[422] Max Jaderberg, Michael Mathieu, Nat McAleese, Nathalie Bradley-Schmieg, Nathaniel Wong, Nicolas Porcel, Steph Hughes-Fitt, Valentin Dalibard, and Wojciech Marian Czarnecki. Open-Ended learning leads to Generally Capable Agents. *Arxiv*, 2107.12808:54, 2021.

[423] Florian Häse, Loïc M. Roch, and Alán Aspuru-Guzik. Chimera: Enabling hierarchy based multi-objective optimization for self-driving laboratories. *ChemRxiv*, 2018.

[424] Florian Häse, Loïc M. Roch, Christoph Kreisbeck, and Alán Aspuru-Guzik. Phoenics: A bayesian optimizer for chemistry. *ACS Central Science*, 4:1134 – 1145, 2018.

[425] Martha M Flores-Leonar, L. M. Mejía-Mendoza, Andrés Aguilar-Granda, Benjamín Sánchez-Lengeling, Hermann Tribukait, Carlos Amador-Bedolla, and Alán Aspuru-Guzik. Materials acceleration platforms: On the way to autonomous experimentation. *Green and Sustainable Chemistry*, 25:100370, 2020.

[426] Sebastian Steiner, Jakob Wolf, Stefan Glatzel, Anna Andreou, Jarosław M. Granda, Graham Keenan, Trevor Hinkley, Gerardo Aragon-Camarasa, Philip J. Kitson, Davide Angelone, and Leroy Cronin. Organic synthesis in a modular robotic system driven by a chemical programming language. *Science*, 363, 2019.

[427] Benjamin P. MacLeod, Fraser G. L. Parlane, Thomas D. Morrissey, Florian Häse, Loïc M. Roch, Kevan E Dettelbach, R. Moreira, Lars P. E. Yunker, Michael Rooney, J. R. Deeth, Veronica Lai, G. J. Ng, Henry Situ, Regan-Heng Zhang, Michael S. Elliott, Ted H. Haley, David J. Dvorak, Alán Aspuru-Guzik, Jason E. Hein, and Curtis P. Berlinguette. Self-driving laboratory for accelerated discovery of thin-film materials. *Science Advances*, 6, 2020.

[428] John J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the United States of America*, 79 8:2554–8, 1982.

[429] German Ignacio Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural networks : the official journal of the International Neural Network Society*, 113: 54–71, 2019.

[430] Alexander Rainer Tassilo Gepperth and Barbara Hammer. Incremental learning algorithms and applications. In *ESANN*, 2016.

[431] Meire Fortunato, Melissa Tan, Ryan Faulkner, Steven Stenberg Hansen, Adrià Puigdomènech Badia, Gavin Buttimore, Charlie Deck, Joel Z. Leibo, and Charles Blundell. Generalization of reinforcement learners with working and episodic memory. In *NeurIPS*, 2019.

[432] Andrea Banino, Adrià Puigdomènech Badia, Raphael Köster, Martin J. Chadwick, Vinícius Flores Zambaldi, Demis Hassabis, Caswell Barry, Matthew M. Botvinick, Dharshan Kumaran, and Charles Blundell. Memo: A deep network for flexible combination of episodic memories. *ArXiv*, abs/2001.10913, 2020.

[433] Tsendsuren Munkhdalai, Alessandro Sordoni, Tong Wang, and Adam Trischler. Metalearned neural memory. In *NeurIPS*, 2019.

[434] Raia Hadsell, Dushyant Rao, Andrei A. Rusu, and Razvan Pascanu. Embracing change: Continual learning in deep neural networks. *Trends in Cognitive Sciences*, 24:1028–1040, 2020.

[435] Hubert Ramsauer, Bernhard Schafl, Johannes Lehner, Philipp Seidl, Michael Widrich, Lukas Gruber, Markus Holzleitner, Milena Pavlovi'c, Geir Kjetil Sandve, Victor Greiff, David P. Kreil, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. Hopfield networks is all you need. *ArXiv*, abs/2008.02217, 2021.

[436] Pentti Kanerva. Sparse distributed memory. 1988.

[437] Yan Wu, Greg Wayne, Alex Graves, and Timothy P. Lillicrap. The kanerva machine: A generative distributed memory. *ArXiv*, abs/1804.01756, 2018.

[438] Adam H. Marblestone, Y. Wu, and Greg Wayne. Product kanerva machines: Factorized bayesian memory. *ArXiv*, abs/2002.02385, 2020.

[439] J. Hawkins, S. Ahmad, S. Purdy, and A. Lavin. *Biological and Machine Intelligence (BAMI)*. 2016. URL https://numenta.com/resources/biological-and-machine-intelligence/.

[440] Yuwei Cui, Alexander Lavin, Subutai Ahmad, and Jeff Hawkins. The htm spatial pooler—a neocortical algorithm for online sparse distributed coding. *Frontiers in Computational Neuroscience*, 11, 2017.

[441] David Wolpert, Christopher P. Kempes, Peter F. Stadler, and Joshua A. Grochow. *The Energetics of Computing in Life and Machines*. The Santa Fe Press, 2019. ISBN 9781947864078.

[442] Kevin Frans, L B Soros, and Olaf Witkowski. Questions for the Open-Ended Evolution Community: Reflections from the 2021 Cross Labs Innovation Science Workshop. *Artificial Life*, 25(1):4, 2021.

[443] Kenneth O. Stanley. Why open-endedness matters. *Artificial Life*, 25:232–235, 2019.

[444] Maxwell Nye, Luke Hewitt, Joshua Tenenbaum, and Armando Solar-Lezama. Learning to infer program sketches. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 4861–4870. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/nye19a.html.

[445] Sumit Gulwani. Automating string processing in spreadsheets using input-output examples. In *POPL '11*, 2011.

[446] Matej Balog, Alexander L. Gaunt, Marc Brockschmidt, Sebastian Nowozin, and Daniel Tarlow. Deepcoder: Learning to write programs. *ArXiv*, abs/1611.01989, 2017.

[447] Justin Gottschlich, Armando Solar-Lezama, Nesime Tatbul, Michael Carbin, Martin Rinard, Regina Barzilay, Saman Amarasinghe, Joshua B. Tenenbaum, and Tim Mattson. The Three Pillars of Machine Programming. In *Proceedings of the 2nd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, MAPL, 2018. ISBN 978-1-4503-5834-7. doi: 10.1145/3211346.3211355. URL http://doi.acm.org/10.1145/3211346.3211355.

[448] Yunming Zhang, Mengjiao Yang, Riyadh Baghdadi, Shoaib Kamil, Julian Shun, and Saman P. Amarasinghe. Graphit - a high-performance dsl for graph analytics. *ArXiv*, abs/1805.00923, 2018.

[449] Riyadh Baghdadi, Jessica Ray, Malek Ben Romdhane, Emanuele Del Sozzo, Abdurrahman Akkas, Yunming Zhang, Patricia Suriana, Shoaib Kamil, and Saman P. Amarasinghe. Tiramisu: A polyhedral compiler for expressing fast and portable code. *2019 IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*, pages 193–205, 2019.

[450] Maaz Bin Safeer Ahmad, Jonathan Ragan-Kelley, Alvin Cheung, and Shoaib Kamil. Automatically Translating Image Processing Libraries to Halide. *ACM Trans. Graph.*, 38(6), nov 2019. ISSN 0730-0301. doi: 10.1145/3355089.3356549. URL https://doi.org/10.1145/3355089.3356549.

[451] Shoaib Kamil, Alvin Cheung, Shachar Itzhaky, and Armando Solar-Lezama. Verified lifting of stencil computations. In *Proceedings of the 37th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI '16, page 711–726, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450342612. doi: 10.1145/2908080.2908117. URL https://doi.org/10.1145/2908080.2908117.

[452] Celine Lee, Justin Gottschlich, and Dan Roth. A Survey on Semantic Parsing for Machine Programming. In *Workshop on Programming Language Processing (PLP) at the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD'21, 2021.

[453] Mejbah Alam, Justin Gottschlich, Nesime Tatbul, Javier S Turek, Tim Mattson, and Abdullah Muzahid. A Zero-Positive Learning Approach for Diagnosing Software Performance Regressions. In H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlchBuc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, NeurIPS 2019, pages 11623–11635. Curran Associates, Inc., 2019. URL http://papers.nips.cc/paper/9337-a-zero-positive-learning-approach-for-diagnosing-software-performance-regressions.pdf.

[454] Ke Li and Jitendra Malik. Learning to optimize. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=ry4Vrt5gl.

[455] Andrew Adams, Karima Ma, Luke Anderson, Riyadh Baghdadi, Tzu-Mao Li, Michaël Gharbi, Benoit Steiner, Steven Johnson, Kayvon Fatahalian, Frédo Durand, and Jonathan Ragan-Kelley. Learning to Optimize Halide with Tree Search and Random Programs. *ACM Trans. Graph.*, 38(4):121:1–121:12, Jul 2019. ISSN 0730-0301. doi: 10.1145/3306346.3322967. URL https://halide-lang.org/papers/halide_autoscheduler_2019.pdf.

[456] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde, Jared Kaplan, Harrison Edwards, Yura Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea. Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, David W. Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William H. Guss, Alex Nichol, Igor Babuschkin, S. Arun Balaji, Shantanu Jain, Andrew Carr, Jan Leike, Joshua Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew M. Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *ArXiv*, abs/2107.03374, 2021.

[457] Kevin Ellis, Daniel Ritchie, Armando Solar-Lezama, and Joshua B. Tenenbaum. Learning to infer graphics programs from hand-drawn images, 2018. URL https://openreview.net/forum?id=H1DJFybC-.

[458] Niranjan Hasabnis and Justin Gottschlich. Controlflag: A self-supervised idiosyncratic pattern detection system for software control structures. In *Proceedings of the 5th ACM SIGPLAN International Symposium on Machine Programming*, MAPS 2021, page 32–42, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450384674. doi: 10.1145/3460945.3464954. URL https://doi.org/10.1145/3460945.3464954.

[459] Rajeev Alur, Rastislav Bodik, Garvit Juniwal, Milo M. K. Martin, Mukund Raghothaman, Sanjit A. Seshia, Rishabh Singh, Armando Solar-Lezama, Emina Torlak, and Abhishek Udupa. Syntax-guided synthesis. In *2013 Formal Methods in Computer-Aided Design*, pages 1–8, 2013. doi: 10.1109/FMCAD.2013.6679385.

[460] Fangke Ye, Shengtian Zhou, Anand Venkat, Ryan Marcus, Nesime Tatbul, Jesmin Jahan Tithi, Niranjan Hasabnis, Paul Petersen, Timothy Mattson, Tim Kraska, Pradeep Dubey, Vivek Sarkar, and Justin Gottschlich. MISIM: A Novel Code Similarity System, 2020.

[461] Nuno Lopes, Farhana Aleen, Abid Muslim Malik, Frank Alexander, Saman P. Amarasinghe, Pavan Balaji, M. Naik, Bill Carlson, Boyana Norris, Barbara Chapman, Swarat Chaudhuri, Madhusudan Parthasarathy, and Krishnan Raghavan. Program synthesis for scientific computing. 2021.

[462] Mejbah Alam, Justin Emile Gottschlich, Nesime Tatbul, Javier Turek, Timothy G. Mattson, and Abdullah Muzahid. A zero-positive learning approach for diagnosing software performance regressions. In *NeurIPS*, 2019.

[463] Mohammad Mejbah Ul Alam and Abdullah Muzahid. Production-run software failure diagnosis via adaptive communication tracking. *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, pages 354–366, 2016.

[464] Sara Hooker. The hardware lottery. *Communications of the ACM*, 64:58 – 65, 2021.

[465] Ge Wang, Jong Chul Ye, and Bruno De Man. Deep learning for tomographic image reconstruction. *Nature Machine Intelligence*, 2(12):737–748, 2020.

[466] Benjamín Sánchez-Lengeling and Alán Aspuru-Guzik. Inverse molecular design using machine learning: Generative models for matter engineering. *Science*, 361:360 – 365, 2018.

[467] Sean Molesky, Zin Lin, Alexander Y. Piggott, Weiliang Jin, Vucković, Jelena , and Alejandro W. Rodriguez. Inverse design in nanophotonics. *Nature Photonics*, 12(11):659–670, October 2018. doi: 10.1038/s41566-018-0246-9.

[468] Sunae So, Trevon Badloe, Jae-Kyo Noh, J. Bravo-Abad, and J. Rho. Deep learning enabled inverse design in nanophotonics. *Nanophotonics*, 9:1041 – 1057, 2020.

[469] Jiaxin Zhang, Sirui Bi, and Guannan Zhang. A directional gaussian smoothing optimization method for computational inverse design in nanophotonics. *Materials & Design*, 197:109213, 2021.

[470] Mérina K Corpinot and Dejan-Krešimir Bučar. A practical guide to the design of molecular crystals. *Crystal Growth & Design*, 19(2):1426–1453, 2018.

[471] Juhwan Noh, Jaehoon Kim, Helge S Stein, Benjamin Sanchez-Lengeling, John M Gregoire, Alan Aspuru-Guzik, and Yousung Jung. Inverse design of solid-state materials via a continuous representation. *Matter*, 1(5):1370–1384, 2019.

[472] Juhwan Noh, Geun Ho Gu, Sungwon Kim, and Yousung Jung. Machine-enabled inverse design of inorganic solid materials: promises and challenges. *Chemical Science*, 11(19):4871–4881, 2020.

[473] Victor Fung, Jiaxin Zhang, Guoxiang Hu, P Ganesh, and Bobby G Sumpter. Inverse design of two-dimensional materials with invertible neural networks. *arXiv preprint arXiv:2106.03013*, 2021.

[474] Muhammad Asim, Max Daniels, Oscar Leong, Ali Ahmed, and Paul Hand. Invertible generative models for inverse problems: mitigating representation error and dataset bias. In *International Conference on Machine Learning*, pages 399–409. PMLR, 2020.

[475] Peter Ertl and Ansgar Schuffenhauer. Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. *Journal of cheminformatics*, 1(1):1–11, 2009.

[476] Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mr Prabhat, and Ryan Adams. Scalable bayesian optimization using deep neural networks. In *International conference on machine learning*, pages 2171–2180. PMLR, 2015.

[477] Nikolaus Hansen, Sibylle D Müller, and Petros Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es). *Evolutionary computation*, 11(1):1–18, 2003.

[478] Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

[479] Jiaxin Zhang, Hoang Tran, Dan Lu, and Guannan Zhang. Enabling long-range exploration in minimization of multimodal functions. *Proceedings of 37th on Uncertainty in Artificial Intelligence (UAI)*.

[480] He Sun and Katherine L Bouman. Deep probabilistic imaging: Uncertainty quantification and multi-modal solution characterization for computational imaging. *arXiv preprint arXiv:2010.14462*, 2020.

[481] Yingxiang Yang, Bo Dai, Negar Kiyavash, and Niao He. Predictive approximate bayesian computation via saddle points. *Proceedings of Machine Learning Research*, 2018.

[482] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8798–8807, 2018.

[483] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. *Advances in neural information processing systems*, 28:3483–3491, 2015.

[484] Francesco Tonolini, Jack Radford, Alex Turpin, Daniele Faccio, and Roderick Murray-Smith. Variational inference for computational imaging inverse problems. *Journal of Machine Learning Research*, 21(179):1–46, 2020.

[485] Jay Whang, Erik Lindgren, and Alex Dimakis. Composing normalizing flows for inverse problems. In *International Conference on Machine Learning*, pages 11158–11169. PMLR, 2021.

[486] Jay Whang, Qi Lei, and Alex Dimakis. Solving inverse problems with a flow-based noise model. In *International Conference on Machine Learning*, pages 11146–11157. PMLR, 2021.

[487] Giannis Daras, Joseph Dean, Ajil Jalal, and Alexandros G Dimakis. Intermediate layer optimization for inverse problems using deep generative models. *arXiv preprint arXiv:2102.07364*, 2021.

[488] Konik Kothari, AmirEhsan Khorashadizadeh, V Maarten, and Ivan Dokmanic. Trumpets: Injective flows for inference and inverse problems. 2021.

[489] Jiaxin Zhang, Victor Fung, and Sirui Bi. Uncertainty-aware image reconstruction with robust generative flows. *AAAI Conference on Artificial Intelligence*, 2022.

[490] Alexander IJ Forrester and Andy J Keane. Recent advances in surrogate-based optimization. *Progress in aerospace sciences*, 45(1-3):50–79, 2009.

[491] Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

[492] Daniel A White, William J Arrighi, Jun Kudo, and Seth E Watts. Multiscale topology optimization using neural network surrogate models. *Computer Methods in Applied Mechanics and Engineering*, 346:1118–1135, 2019.

[493] Tianju Xue, Alex Beatson, Sigrid Adriaenssens, and Ryan Adams. Amortized finite element analysis for fast pde-constrained optimization. In *International Conference on Machine Learning*, pages 10638–10647. PMLR, 2020.

[494] David Brookes, Hahnbeom Park, and Jennifer Listgarten. Conditioning by adaptive sampling for robust design. In *International Conference on Machine Learning*, pages 773–782. PMLR, 2019.

[495] Simiao Ren, Willie Padilla, and Jordan Malof. Benchmarking deep inverse models over time, and the neural-adjoint method. *arXiv preprint arXiv:2009.12919*, 2020.

[496] Yang Deng, Simiao Ren, Kebin Fan, Jordan M Malof, and Willie J Padilla. Neural-adjoint method for the inverse design of all-dielectric metasurfaces. *Optics Express*, 29(5):7526–7534, 2021.

[497] Omar Khatib, Simiao Ren, Jordan Malof, and Willie J Padilla. Deep learning the electromagnetic properties of metamaterials—a comprehensive review. *Advanced Functional Materials*, page 2101748, 2021.

[498] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *arXiv preprint arXiv:1605.08803*, 2016.

[499] Diederik P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *arXiv preprint arXiv:1807.03039*, 2018.

[500] Will Grathwohl, Ricky TQ Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367*, 2018.

[501] Hao Wu, Jonas Köhler, and Frank Noé. Stochastic normalizing flows. *arXiv preprint arXiv:2002.06707*, 2020.

[502] Didrik Nielsen, Priyank Jaini, Emiel Hoogeboom, Ole Winther, and Max Welling. Survae flows: Surjections to bridge the gap between vaes and flows. *Advances in Neural Information Processing Systems*, 33, 2020.

[503] Lynton Ardizzone, Jakob Kruse, Carsten Rother, and Ullrich Köthe. Analyzing inverse problems with invertible neural networks. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJed6j0cKX.

[504] Lynton Ardizzone, Carsten Lüth, Jakob Kruse, Carsten Rother, and Ullrich Köthe. Guided image generation with conditional invertible neural networks. *arXiv preprint arXiv:1907.02392*, 2019.

[505] Jakob Kruse, Lynton Ardizzone, Carsten Rother, and Ullrich Köthe. Benchmarking invertible architectures on inverse problems. *arXiv preprint arXiv:2101.10763*, 2021.

[506] Yang Deng, Simiao Ren, Kebin Fan, Jordan M Malof, and Willie J Padilla. Neural-adjoint method for the inverse design of all-dielectric metasurfaces. *arXiv preprint arXiv:2012.05020*, 2020.

[507] Jiaxin Zhang, Victor Fung, and Sirui Bi. Accelerating inverse design via intelligent priors with space-filling sampling. *International Conference on Artificial Intelligence and Statistics*, 2021.

[508] Michael D Shields and Jiaxin Zhang. The generalization of latin hypercube sampling. *Reliability Engineering & System Safety*, 148:96–108, 2016.

[509] G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, Naftali Tishby, Leslie Vogt-Maranto, and Lenka Zdeborov'a. Machine learning and the physical sciences. *Reviews of Modern Physics*, 91:045002, 2019.

[510] Jiaxin Zhang and Michael D Shields. On the quantification and efficient propagation of imprecise probabilities resulting from small datasets. *Mechanical Systems and Signal Processing*, 98:465–483, 2018.

[511] Jiaxin Zhang and Michael D Shields. The effect of prior probabilities on quantification and propagation of imprecise probabilities resulting from small datasets. *Computer Methods in Applied Mechanics and Engineering*, 334:483–506, 2018.

[512] Jiaxin Zhang and Michael Shields. On the quantification and efficient propagation of imprecise probabilities with copula dependence. *International Journal of Approximate Reasoning*, 122:24–46, 2020.

[513] Lynton Ardizzone, Jakob Kruse, Sebastian J. Wirkert, D. Rahner, E. Pellegrini, R. Klessen, L. Maier-Hein, C. Rother, and U. Köthe. Analyzing inverse problems with invertible neural networks. *ArXiv*, abs/1808.04730, 2019.

[514] Hector Zenil, Narsis A Kiani, Allan A Zea, and Jesper Tegnér. Causal deconvolution by algorithmic generative models. *Nature Machine Intelligence*, 1(1), 2019. ISSN 25225839. doi: 10.1038/s42256-018-0005-0.

[515] J. Neumann. Probabilistic logic and the synthesis of reliable organisms from unreliable components. 1956.

[516] Yinhao Zhu and N. Zabaras. Bayesian deep convolutional encoder-decoder networks for surrogate modeling and uncertainty quantification. *ArXiv*, abs/1801.06879, 2018.

[517] R. Dandekar, Vaibhav Dixit, Mohamed Tarek, Aslan Garcia-Valadez, and C. Rackauckas. Bayesian neural ordinary differential equations. *ArXiv*, abs/2012.07244, 2020.

[518] Y. Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. *ArXiv*, abs/1506.02142, 2016.

[519] Henri Poincaré. Calcul des probabilités. 1896.

[520] Chris J. Oates and Timothy John Sullivan. A modern retrospective on probabilistic numerics. *Stat. Comput.*, 29:1335–1351, 2019.

[521] A. Scibior, Zoubin Ghahramani, and Andrew D. Gordon. Practical probabilistic programming with monads. *Proceedings of the 2015 ACM SIGPLAN Symposium on Haskell*, 2015.

[522] Owain Evans, Andreas Stuhlmüller, John Salvatier, and Daniel Filan. Modeling Agents with Probabilistic Programs. http://agentmodels.org, 2017. Accessed: 2021-8-31.

[523] Yura N. Perov, L. Graham, Kostis Gourgoulias, Jonathan G Richens, Ciarán M. Lee, Adam Baker, and Saurabh Johri. Multiverse: Causal reasoning using importance sampling in probabilistic programming. *ArXiv*, abs/1910.08091, 2019.

[524] Eli Bingham, Jonathan P. Chen, Martin Jankowiak, Fritz Obermeyer, Neeraj Pradhan, Theofanis Karaletsos, Rohit Singh, Paul A. Szerlip, Paul Horsfall, and Noah D. Goodman. Pyro: Deep universal probabilistic programming. *J. Mach. Learn. Res.*, 20:28:1–28:6, 2019.

[525] John Salvatier, Thomas V. Wiecki, and Christopher J Fonnesbeck. Probabilistic programming in python using pymc3. *PeerJ Comput. Sci.*, 2:e55, 2016.

[526] Marco F. Cusumano-Towner, Feras A. Saad, Alexander K. Lew, and Vikash K. Mansinghka. Gen: a general-purpose probabilistic programming system with programmable inference. *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation*, 2019.

[527] Santiago Hernández-Orozco, Hector Zenil, Jürgen Riedel, Adam Uccello, Narsis Aftab Kiani, and Jesper Tegnér. Algorithmic probability-guided machine learning on non-differentiable spaces. *Frontiers in Artificial Intelligence*, 3, 2020.

[528] David M. Blei. Build, compute, critique, repeat: Data analysis with latent variable models. 2014.

[529] Zenna Tavares, Javier Burroni, Edgar Minaysan, Armando Solar-Lezama, and R. Ranganath. Soft constraints for inference with declarative knowledge. *ArXiv*, abs/1901.05437, 2019.

[530] S. Witty, Alexander K. Lew, David D. Jensen, and Vikash K. Mansinghka. Bayesian causal inference via probabilistic program synthesis. *ArXiv*, abs/1910.14124, 2019.

[531] Zenna Tavares, James Koppel, X. Zhang, and Armando Solar-Lezama. A language for counterfactual generative models. 2019.

[532] Olavi Nevanlinna. Remarks on picard-lindelöf iteration. *BIT Numerical Mathematics*, 29:535–562, 1989.

[533] Joris M. Mooij, Dominik Janzing, and Bernhard Schölkopf. From ordinary differential equations to structural causal models: the deterministic case. *ArXiv*, abs/1304.7920, 2013.

[534] Ken ichi Funahashi and Yuichi Nakamura. Approximation of dynamical systems by continuous time recurrent neural networks. *Neural Networks*, 6:801–806, 1993.

[535] Charles Vorbach, Ramin M. Hasani, Alexander Amini, Mathias Lechner, and Daniela Rus. Causal navigation by continuous-time neural networks. *ArXiv*, abs/2106.08314, 2021.

[536] Ramin M. Hasani, Mathias Lechner, Alexander Amini, Daniela Rus, and Radu Grosu. Liquid time-constant networks. In *AAAI*, 2021.

[537] Mathias Lechner, Ramin M. Hasani, Alexander Amini, Thomas A. Henzinger, Daniela Rus, and Radu Grosu. Neural circuit policies enabling auditable autonomy. *Nature Machine Intelligence*, 2:642–652, 2020.

[538] H. Zenil, N. A. Kiani, F. S Abrahão, and J. N. Tegnér. Algorithmic Information Dynamics. *Scholarpedia*, 15(7):53143, 2020. doi: 10.4249/scholarpedia.53143. revision #195807.

[539] Rainer Herges. Coarctate transition states: the discovery of a reaction principle. *J. Chem. Inf. Comput. Sci.*, 34:91–102, 1994.

[540] John Bradshaw, Matt J. Kusner, Brooks Paige, Marwin H. S. Segler, and José Miguel Hernández-Lobato. A generative model for electron paths. *arXiv: Chemical Physics*, 2019.

[541] Connor W. Coley, Regina Barzilay, T. Jaakkola, William H. Green, and Klavs F. Jensen. Prediction of organic reaction outcomes using machine learning. *ACS Central Science*, 3:434 – 443, 2017.

[542] Wengong Jin, Connor W. Coley, Regina Barzilay, and T. Jaakkola. Predicting organic reaction outcomes with weisfeiler-lehman network. In *NIPS*, 2017.

[543] Marwin H. S. Segler and Mark P. Waller. Neural-symbolic machine learning for retrosynthesis and reaction prediction. *Chemistry*, 23 25:5966–5971, 2017.

[544] Jennifer N. Wei, David Kristjanson Duvenaud, and Alán Aspuru-Guzik. Neural networks for the prediction of organic chemistry reactions. *ACS Central Science*, 2:725 – 732, 2016.

[545] Alpha Albert Lee, Qingyi Yang, Vishnu Sresht, Peter Bolgar, Xinjun Hou, Jacquelyn L. Klug-McLeod, and Christopher R. Butler. Molecular transformer unifies reaction prediction and retrosynthesis across pharma chemical space. *Chemical communications*, 2019.

[546] Charles A. Holt and Alvin E. Roth. The nash equilibrium: A perspective. *Proceedings of the National Academy of Sciences of the United States of America*, 101:3999 – 4002, 2004.

[547] Kurt L. M. Drew, Hakim Baiman, Prashannata Khwaounjoo, Bo Yu, and Jóhannes Reynisson. Size estimation of chemical space: how big is it? *Journal of Pharmacy and Pharmacology*, 64, 2012.

[548] Juan-Pablo Correa-Baena, Kedar Hippalgaonkar, Jeroen van Duren, Shaffiq A. Jaffer, Vijay Ramaseshan Chandrasekhar, Vladan Stevanović, Cyrus Wadia, Supratik Guha, and Tonio Buonassisi. Accelerating materials development via automation, machine learning, and high-performance computing. *ArXiv*, abs/1803.11246, 2018.

[549] Benjamin Sanchez-Lengeling and Alán Aspuru-Guzik. Inverse molecular design using machine learning: Generative models for matter engineering. *Science*, 361(6400):360–365, 2018.

[550] Loïc M. Roch, Florian Häse, Christoph Kreisbeck, Teresa Tamayo-Mendoza, Lars P. E. Yunker, Jason E. Hein, and Alán Aspuru-Guzik. Chemos: An orchestration software to democratize autonomous discovery. *PLoS ONE*, 15, 2020.

[551] P. Dayan. Helmholtz machines and wake-sleep learning. 2000.

[552] Z. Ballard, C. Brown, A.M. Madni, and et al. Machine learning and computation-enabled intelligent sensor design. *Nature Machine Intelligence*, 3:556–565, 2021.

[553] Nicolaas M. Angenent-Mari, Alexander S Garruss, L. Soenksen, G. Church, and J. J. Collins. A deep learning approach to programmable rna switches. *Nature Communications*, 11, 2020.

[554] Azalia Mirhoseini, Anna Goldie, Mustafa Yazgan, Joe Wenjie Jiang, Ebrahim M. Songhori, Shen Wang, Young-Joon Lee, Eric Johnson, Omkar Pathak, Azade Nazi, Jiwoo Pak, Andy Tong, Kavya Srinivasa, Will Hang, Emre Tuncer, Quoc V. Le, James Laudon, Richard Ho, Roger Carpenter, and Jeff Dean. A graph placement methodology for fast chip design. *Nature*, 594 7862:207–212, 2021.

[555] Logan G. Wright, Tatsuhiro Onodera, Martin M. Stein, Tianyu Wang, Darren T. Schachter, Zoey Hu, and Peter Leonard McMahon. Deep physical neural networks enabled by a backpropagation algorithm for arbitrary physical systems. *ArXiv*, abs/2104.13386, 2021.

[556] E. Horvitz and Tim Paek. Complementary computing: policies for transferring callers from dialog systems to human receptionists. *User Modeling and User-Adapted Interaction*, 17:159–182, 2007.

[557] Ece Kamar, Severin Hacker, and E. Horvitz. Combining human and machine intelligence in large-scale crowdsourcing. In *AAMAS*, 2012.

[558] Maithra Raghu, K. Blumer, Greg S Corrado, J. Kleinberg, Z. Obermeyer, and S. Mullainathan. The algorithmic automation problem: Prediction, triage, and human effort. *ArXiv*, abs/1903.12220, 2019.

[559] Burr Settles. Active learning. In *Active Learning*, 2012.

[560] Peiyun Hu, Zachary Chase Lipton, Anima Anandkumar, and D. Ramanan. Active learning with partial feedback. *ArXiv*, abs/1802.07427, 2019.

[561] B. Wilder, E. Horvitz, and Ece Kamar. Learning to complement humans. In *IJCAI*, 2020.

[562] M. Sensoy, Melih Kandemir, and Lance M. Kaplan. Evidential deep learning to quantify classification uncertainty. In *NeurIPS*, 2018.

[563] Nis Meinert and Alexander Lavin. Multivariate deep evidential regression. *ArXiv*, abs/2104.06135, 2021.

[564] Audun Jøsang. Subjective logic: A formalism for reasoning under uncertainty. 2016.

[565] Kate Goddard, Abdul V. Roudsari, and Jeremy C. Wyatt. Automation bias: a systematic review of frequency, effect mediators, and mitigators. *Journal of the American Medical Informatics Association : JAMIA*, 19 1:121–7, 2012.

[566] Richard J. Tomsett, Alun David Preece, Dave Braines, Federico Cerutti, Supriyo Chakraborty, Mani B. Srivastava, Gavin Pearson, and Lance M. Kaplan. Rapid trust calibration through interpretable and uncertainty-aware ai. *Patterns*, 1, 2020.

[567] John C. Doyle. Guaranteed margins for lqg regulators. *IEEE Transactions on Automatic Control*, 23:756–757, 1978.

[568] Ian R. Petersen and Roberto Tempo. Robust control of uncertain systems: Classical results and recent developments. *Autom.*, 50:1315–1335, 2014.

[569] Zhiming Zhao. "data intensive sciences in high performance computing" panel at the 2015 international conference on high performance computing & simulation (hpcs 2015). 2015.

[570] David A. Humphreys, Ana Kupresanin, Mark D. Boyer, J. M. Canik, C. S. Chang, Eric C. Cyr, R. S. Granetz, Jeffrey A. F. Hittinger, Egemen Kolemen, Earl Christopher Lawrence, Valerio Pascucci, Anggi Patra, and David P. Schissel. Advancing fusion with machine learning research needs workshop report. *Journal of Fusion Energy*, 39:123–155, 2020.

[571] John L. Schnase, Tsengdar J. Lee, Chris Mattmann, Christopher Lynnes, Luca Cinquini, Maglio Paul, Ramire, Andrew F. Hart, Dean N. Williams, Duane E. Waliser, Pamela Livingstone Rinsland, William Webster, Daniel Q. Duffy, Mark A. McInerney, Glenn S. Tamkin, Gerald Potter, and Laura Carrier. Big data challenges in climate science. 2015.

[572] John L. Schnase, Daniel Q. Duffy, Glenn S. Tamkin, Denis Nadeau, John H. Thompson, Christina M. Grieg, Mark A. McInerney, and William Webster. Merra analytic services: Meeting the big data challenges of climate science through cloud-enabled climate analytics-as-a-service. *Comput. Environ. Urban Syst.*, 61:198–211, 2017.

[573] Eliu A. Huerta, Asad Khan, Edward Davis, Colleen Bushell, William Gropp, Daniel S. Katz, Volodymyr V. Kindratenko, Seid Koric, William T. C. Kramer, Brendan McGinty, Kenton McHenry, and Aaron Saxton. Convergence of artificial intelligence and high performance computing on nsf-supported cyberinfrastructure. *Journal of Big Data*, 7:1–12, 2020.

[574] Lauri Himanen, Amber Geurts, Adam Stuart Foster, and Patrick Rinke. Data-driven materials science: Status, challenges, and perspectives. *Advanced Science*, 6, 2019.

[575] Benjamin J. Blaiszik, Logan T. Ward, Marcus Schwarting, Jonathon Gaff, Ryan Chard, Danie Webster Pike, Kyle Chard, and Ian T. Foster. A data ecosystem to support machine learning in materials science. *MRS Communications*, 2019.

[576] Sander Janssen, Cheryl H. Porter, Andrew D. Moore, Ioannis N. Athanasiadis, Ian T. Foster, James W. Jones, and John Antle. Towards a new generation of agricultural system data, models and knowledge products: Information and communication technology. *Agricultural Systems*, 155:200 – 212, 2017.

[577] Mark D. Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Olavo Bonino da Silva Santos, Philip E. Bourne, Jildau Bouwman, Anthony J. Brookes, Tim Clark, Mercè Crosas, Ingrid Dillo, Olivier Dumon, Scott C. Edmunds, Chris T. A. Evelo, Richard Finkers, Alejandra N. González-Beltrán, Alasdair J. G. Gray, Paul Groth, Carole A. Goble, Jeffrey S. Grethe, Jaap Heringa, Peter A. C. 't Hoen, Rob W. W. Hooft, Tobias Kuhn, Ruben G. Kok, Joost N. Kok, Scott J. Lusher, Maryann E. Martone, Albert Mons, Abel Laerte Packer, Bengt Persson, Philippe Rocca-Serra, Marco Roos, René C van Schaik, Susanna-Assunta Sansone, Erik Anthony Schultes, Thierry Sengstag, Ted Slater, George O. Strawn, Morris A. Swertz, Mark Thompson, Johan van der Lei, Erik M van Mulligen, Jan Velterop, Andra Waagmeester, Peter Wittenburg, Katy Wolstencroft, Jun Zhao, and Barend Mons. The fair guiding principles for scientific data management and stewardship. *Scientific Data*, 3, 2016.

[578] Moritz Lehne, Julian Sass, Andrea Essenwanger, Josef Schepers, and Sylvia Thun. Why digital medicine depends on interoperability. *NPJ Digital Medicine*, 2, 2019.

[579] Goksel Misirli, Jennifer S. Hallinan, Matthew R. Pocock, Phillip W. Lord, James Alastair McLaughlin, Herbert M. Sauro, and Anil Wipat. Data integration and mining for synthetic biology design. *ACS synthetic biology*, 5 10:1086–1097, 2016.

[580] Shazia Afzal, C Rajmohan, Manish Kesarwani, Sameep Mehta, and Hima Patel. Data readiness report. *2021 IEEE International Conference on Smart Data Services (SMDS)*, pages 42–51, 2021.

[581] Neil Lawrence. Data readiness levels. *ArXiv*, abs/1705.02245, 2017.

[582] Susan B. Davidson and Juliana Freire. Provenance and scientific workflows: challenges and opportunities. In *SIGMOD Conference*, 2008.

[583] Jingwei Huang. From big data to knowledge: Issues of provenance, trust, and scientific computing integrity. *2018 IEEE International Conference on Big Data (Big Data)*, pages 2197–2205, 2018.

[584] Konstantinos Bousmalis, Alex Irpan, Paul Wohlhart, Yunfei Bai, Matthew Kelcey, Mrinal Kalakrishnan, Laura Downs, Julian Ibarz, Peter Pastor, Kurt Konolige, Sergey Levine, and Vincent Vanhoucke. Using simulation and domain adaptation to improve efficiency of deep robotic grasping. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4243–4250, 2018.

[585] Matthew Johnson-Roberson, Charlie Barto, Rounak Mehta, Sharath Nittur Sridhar, Karl Rosaen, and Ram Vasudevan. Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks? *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 746–753, 2017.

[586] Hoo-Chang Shin, Neil A. Tenenholtz, Jameson K. Rogers, Christopher G. Schwarz, Matthew L. Senjem, Jeffrey L. Gunter, Katherine P. Andriole, and Mark H. Michalski. Medical image synthesis for data augmentation and anonymization using generative adversarial networks. *ArXiv*, abs/1807.10225, 2018.

[587] Nazmiye Ceren Abay, Yan Zhou, Murat Kantarcioglu, Bhavani M. Thuraisingham, and Latanya Sweeney. Privacy preserving synthetic data release using deep learning. In *ECML/PKDD*, 2018.

[588] Amirsina Torfi, Edward A. Fox, and Chandan K. Reddy. Differentially private synthetic medical data generation using convolutional gans. *ArXiv*, abs/2012.11774, 2020.

[589] Jan Pablo Burgard, Jan-Philipp Kolb, Hariolf Merkle, and Ralf T. Münnich. Synthetic data for open and reproducible methodological research in social sciences and official statistics. *AStA Wirtschafts- und Sozialstatistisches Archiv*, 11:233–244, 2017.

[590] Kamil Deja, Tomasz Trzciński, and Łukasz Kamil Graczykowski. Generative models for fast cluster simulations in the tpc for the alice experiment. *Advances in Intelligent Systems and Computing*, 2019.

[591] Joseph DeRose, Risa H. Wechsler, Matthew Becker, Michael T. Busha, Eli S. Rykoff, Niall MacCrann, Brandon M. S. Erickson, August E. Evrard, Andrey V. Kravtsov, Daniel Gruen, Sahar Allam, Santiago Ávila, Sarah Bridle, D. Brooks, Elizabeth Buckley-Geer, Aurelio Carnero Rosell, Matias Carrasco Kind, Jorge Carretero, Francisco J. Castander, R. Cawthon, Martín Crocce, Luiz Nicolaci da Costa, Christopher Davis, Juan de Vicente, Jorg P. Dietrich, Peter Doel, Alex Drlica-Wagner, Pablo Fosalba, Joshua A. Frieman, Juan García-Bellido, Gaston R. Gutiérrez, William G. Hartley, Devon L. Hollowood, Ben Hoyle, David J. James, E. Krause, Kyler W. Kuehn, Nikolay Kuropatkin, Marcos Lima, Marcio A. G. Maia, Felipe Menanteau, Christopher J. Miller, Ramon Miquel, R. L. C. Ogando, Andr'es Plazas Malag'on, A. K. Romer, Eusebio Sánchez, Rafe H. Schindler, Santiago Serrano, I. Sevilla-Noarbe, Mathew Smith, Eric Suchyta, Mollye E. C. Swanson, Gregory G. Tarlé, and Vinu Vikram. The buzzard flock: Dark energy survey synthetic sky catalogs. *arXiv: Cosmology and Nongalactic Astrophysics*, 2019.

[592] Debbie Rankin, Michaela M. Black, Raymond R. Bond, Jonathan G. Wallace, Maurice D. Mulvenna, and Gorka Epelde. Reliability of supervised machine learning using synthetic data in health care: Model to preserve privacy for data sharing. *JMIR Medical Informatics*, 8, 2020.

[593] Amlan Kar, Aayush Prakash, Ming-Yu Liu, Eric Cameracci, Justin Yuan, Matt Rusiniak, David Acuna, Antonio Torralba, and Sanja Fidler. Meta-sim: Learning to generate synthetic datasets. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4550–4559, 2019.

[594] Stephan Rabanser, Stephan Günnemann, and Zachary Chase Lipton. Failing loudly: An empirical study of methods for detecting dataset shift. In *NeurIPS*, 2019.

[595] Tong Meng, Xuyang Jing, Zheng Yan, and Witold Pedrycz. A survey on machine learning for data fusion. *Inf. Fusion*, 57: 115–129, 2020.

[596] Xin Dong, Laure Berti-Équille, and Divesh Srivastava. Data fusion: Resolving conflicts from multiple sources. In *Handbook of Data Quality*, 2013.

[597] Jean Claude Burgelman, Corina Pascu, Katarzyna Szkuta, René von Schomberg, Athanasios Karalopoulos, Konstantinos Repanas, and Michel Schouppe. Open science, open data, and open scholarship: European policies to make science fit for the twenty-first century. *Frontiers in Big Data*, 2, 2019.

[598] Ben Hutchinson, Andrew Smart, A. Hanna, Emily L. Denton, Christina Greer, Oddur Kjartansson, Parker Barnes, and Margaret Mitchell. Towards accountability for machine learning datasets: Practices from software engineering and infrastructure. *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, 2021.

[599] Alexander M. Grannan, Kanika Sood, Boyana Norris, and Anshu Dubey. Understanding the landscape of scientific software used on high-performance computing platforms. *The International Journal of High Performance Computing Applications*, 34: 465 – 477, 2020.

[600] Ann L. Chervenak, Ian T. Foster, Carl Kesselman, Charles A. Salisbury, and Steven Tuecke. The data grid: Towards an architecture for the distributed management and analysis of large scientific datasets. *J. Netw. Comput. Appl.*, 23:187–200, 2000.

[601] Yogesh L. Simmhan, Catharine van Ingen, Alexander S. Szalay, Roger S. Barga, and James N. Heasley. Building reliable data pipelines for managing community data using scientific workflows. *2009 Fifth IEEE International Conference on e-Science*, pages 321–328, 2009.

[602] Ben Blamey, Salman Zubair Toor, Martin Dahlö, Håkan Wieslander, Philip J Harrison, Ida-Maria Sintorn, Alan Sabirsh, Carolina Wählby, Ola Spjuth, and Andreas Hellander. Rapid development of cloud-native intelligent data pipelines for scientific data streams using the haste toolkit. *bioRxiv*, 2020.

[603] Ewa Deelman, Tom Peterka, Ilkay Altintas, Christopher D. Carothers, Kerstin Kleese van Dam, Kenneth Moreland, M. Parashar, Lavanya Ramakrishnan, Michela Taufer, and Jeffrey S. Vetter. The future of scientific workflows. *The International Journal of High Performance Computing Applications*, 32:159 – 175, 2018.

[604] Sandro Fiore, Mohamed Bakhouya, and Waleed W. Smari. On the road to exascale: Advances in high performance computing and simulations - an overview and editorial. *Future Gener. Comput. Syst.*, 82:450–458, 2018.

[605] Wo L. Chang, David Boyd, and Orit Levin. Nist big data interoperability framework: Volume 6, reference architecture. 2019.

[606] Jan-Jo. Chen, Alok N. Choudhary, Steven D. Feldman, Bruce Hendrickson, C. R. Johnson, Richard P. Mount, Vivek Sarkar, Vicky White, and Dean Williams. Synergistic challenges in data-intensive science and exascale computing: Doe ascac data subcommittee report. 2013.

[607] Mohand-Said Mezmaz, Nouredine Melab, and El-Ghazali Talbi. A grid-enabled branch and bound algorithm for solving challenging combinatorial optimization problems. *2007 IEEE International Parallel and Distributed Processing Symposium*, pages 1–9, 2007.

[608] Nouredine Melab, Jan Gmys, Mohand-Said Mezmaz, and Daniel Tuyttens. Multi-core versus many-core computing for many-task branch-and-bound applied to big optimization problems. *Future Gener. Comput. Syst.*, 82:472–481, 2018.

[609] Guillaume Oger, David Le Touzé, David Guibert, Matthieu De Leffe, John Biddiscombe, Jérome Soumagne, and Jean-Guillaume Piccinali. On distributed memory mpi-based parallelization of sph codes in massive hpc context. *Comput. Phys. Commun.*, 200:1–14, 2016.

[610] The Enzo Collaboration Greg L. Bryan, Michael L. Norman, Brian W. O'Shea, Tom Abel, John H. Wise, Matthew J. Turk, Daniel R. Reynolds, D. Collins, Peng Wang, Samuel W. Skillman, Britton D. Smith, Robert Harkness, James Bordner, Ji hoon Kim, Michael Kuhlen, Haoting Xu, Nathan J. Goldbaum, Cameron B. Hummels, Alexei G. Kritsuk, Elizabeth J. Tasker, S. A. Skory, Christine M. Simpson, Oliver Hahn, Jeffrey S. Oishi, Geoffrey C. So, Fen Zhao, Renyue Cen, and Yuan Li. Enzo: An adaptive mesh refinement code for astrophysics. *Astrophysical Journal Supplement Series*, 211:19, 2014.

[611] Philip F. Hopkins. A new class of accurate, mesh-free hydrodynamic simulation methods. *Monthly Notices of the Royal Astronomical Society*, 450:53–110, 2015.

[612] Satish Balay, Shrirang Abhyankar, Mark F. Adams, Steven Benson, Jed Brown, Peter Brune, Kris Buschelman, Emil Constantinescu, Lisandro Dalcin, Alp Dener, Victor Eijkhout, William D. Gropp, Václav Hapla, Tobin Isaac, Pierre Jolivet, Dmitry Karpeev, Dinesh Kaushik, Matthew G. Knepley, Fande Kong, Scott Kruger, Dave A. May, Lois Curfman McInnes, Richard Tran Mills, Lawrence Mitchell, Todd Munson, Jose E. Roman, Karl Rupp, Patrick Sanan, Jason Sarich, Barry F. Smith, Stefano Zampini, Hong Zhang, Hong Zhang, and Junchao Zhang. PETSc/TAO users manual. Technical Report ANL-21/39 - Revision 3.16, Argonne National Laboratory, 2021.

[613] Thorsten Kurth, Sean Treichler, Josh Romero, Mayur Mudigonda, Nathan Luehr, Everett H. Phillips, Ankur Mahesh, Michael A. Matheson, Jack R. Deslippe, Massimiliano Fatica, Prabhat, and Michael Houston. Exascale deep learning for climate analytics. *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 649–660, 2018.

[614] Dan Lu and Daniel M. Ricciuto. Efficient surrogate modeling methods for large-scale earth system models based on machine learning techniques. *ArXiv*, abs/1901.05125, 2019.

[615] David E. Shaw, J.P. Grossman, Joseph A. Bank, Brannon Batson, J. Adam Butts, Jack C. Chao, Martin M. Deneroff, Ron O. Dror, Amos Even, Christopher H. Fenton, Anthony Forte, Joseph Gagliardo, Gennette Gill, Brian Greskamp, C. Richard Ho, Douglas J. Ierardi, Lev Iserovich, Jeffrey S. Kuskin, Richard H. Larson, Timothy Layman, Li-Siang Lee, Adam K. Lerer, Chester Li, Daniel Killebrew, Kenneth M. Mackenzie, Shark Yeuk-Hai Mok, Mark A. Moraes, Rolf Mueller, Lawrence J. Nociolo, Jon L. Peticolas, Terry Quan, Daniel Ramot, John K. Salmon, Daniele P. Scarpazza, U. Ben Schafer, Naseer Siddique, Christopher W. Snyder, Jochen Spengler, Ping Tak Peter Tang, Michael Theobald, Horia Toma, Brian Towles, Benjamin Vitale, Stanley C. Wang, and Cliff Young. Anton 2: Raising the bar for performance and programmability in a special-purpose molecular dynamics supercomputer. In *SC '14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 41–53, 2014. doi: 10.1109/SC.2014.9.

[616] Alexander Brace, Michael Salim, Vishal Subbiah, Heng Ma, Murali Emani, Anda Trifa, Austin R. Clyde, Corey Adams, Thomas Uram, Hyunseung Yoo, Andew Hock, Jessica Liu, Venkatram Vishwanath, and Arvind Ramanathan. Stream-ai-md: Streaming ai-driven adaptive molecular simulations for heterogeneous computing platforms. In *Proceedings of the Platform for Advanced Scientific Computing Conference*, PASC '21, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450385633. doi: 10.1145/3468267.3470578. URL https://doi.org/10.1145/3468267.3470578.

[617] Kamil Rocki, Dirk Van Essendelft, Ilya Sharapov, Robert Schreiber, Michael Morrison, Vladimir Kibardin, Andrey Portnoy, Jean Francois Dietiker, Madhava Syamlal, and Michael James. *Fast Stencil-Code Computation on a Wafer-Scale Processor*. IEEE Press, 2020. ISBN 9781728199986.

[618] Stefan M. Larson, Christopher D. Snow, Michael R. Shirts, and Vijay S. Pande. Folding@home and genome@home: Using distributed computing to tackle previously intractable problem. 2009.

[619] Scott D. Stoller, Michael Carbin, Sarita V. Adve, Kunal Agrawal, Guy E. Blelloch, Dan, Stanzione, Katherine A. Yelick, and Matei A. Zaharia. Future directions for parallel and distributed computing: Spx 2019 workshop report. 2019.

[620] Sébastien Rumley, Meisam Bahadori, Robert P. Polster, Simon David Hammond, David M. Calhoun, Ke Wen, Arun Rodrigues, and Keren Bergman. Optical interconnects for extreme scale computing systems. *Parallel Comput.*, 64:65–80, 2017.

[621] Qixiang Cheng, Meisam Bahadori, Madeleine Strom Glick, Sébastien Rumley, and Keren Bergman. Recent advances in optical technologies for data centers: a review. *Optica*, 2018.

[622] Nora Abi Akar, Benjamin Cumming, Vasileios Karakasis, Anne Küsters, Wouter Klijn, Alexander Peyser, and Stuart Yates. Arbor — a morphologically-detailed neural network simulation library for contemporary high-performance computing architectures. *2019 27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, pages 274–282, 2019.

[623] Cédric Renggli, Luka Rimanic, Nezihe Merve Gurel, Bojan Karlavs, Wentao Wu, and Ce Zhang. A data quality-driven view of mlops. *ArXiv*, abs/2102.07750, 2021.

[624] Thomas Häner, Damian S Steiger, Mikhail Smelyanskiy, and Matthias Troyer. High performance emulation of quantum circuits. In *SC'16: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 866–874. IEEE, 2016.

[625] Mikhail Smelyanskiy, Nicolas PD Sawaya, and Alán Aspuru-Guzik. qhipster: The quantum high performance software testing environment. *arXiv preprint arXiv:1601.07195*, 2016.

[626] Benjamin Villalonga, Dmitry Lyakh, Sergio Boixo, Hartmut Neven, Travis S Humble, Rupak Biswas, Eleanor G Rieffel, Alan Ho, and Salvatore Mandrà. Establishing the quantum supremacy frontier with a 281 pflop/s simulation. *Quantum Science and Technology*, 5(3):034003, 2020.

[627] Yong (Alexander) Liu, Xin (Lucy) Liu, Fang (Nancy) Li, Haohuan Fu, Yuling Yang, Jiawei Song, Pengpeng Zhao, Zhen Wang, Dajia Peng, Huarong Chen, Chu Guo, Heliang Huang, Wenzhao Wu, and Dexun Chen. Closing the "quantum supremacy" gap: Achieving real-time simulation of a random quantum circuit using a new sunway supercomputer. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '21, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450384421. doi: 10.1145/3458817.3487399. URL https://doi.org/10.1145/3458817.3487399.

[628] Xin Liu, Chu Guo, Yong Liu, Yuling Yang, Jiawei Song, Jie Gao, Zhen Wang, Wenzhao Wu, Dajia Peng, Pengpeng Zhao, et al. Redefining the quantum supremacy baseline with a new generation sunway supercomputer. *arXiv:2111.01066*, 2021.

[629] RP Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6), 1982.

[630] John Preskill. Quantum computing in the nisq era and beyond. *Quantum*, 2:79, 2018.

[631] N Cody Jones, Rodney Van Meter, Austin G Fowler, Peter L McMahon, Jungsang Kim, Thaddeus D Ladd, and Yoshihisa Yamamoto. Layered architecture for quantum computing. *Physical Review X*, 2(3):031007, 2012.

[632] Giuseppe Carleo and Matthias Troyer. Solving the quantum many-body problem with artificial neural networks. *Science*, 355 (6325):602–606, 2017.

[633] Giacomo Torlai, Guglielmo Mazzola, Juan Carrasquilla, Matthias Troyer, Roger Melko, and Giuseppe Carleo. Neural-network quantum state tomography. *Nature Physics*, 14(5):447–450, 2018.

[634] Juan Carrasquilla. Machine learning for quantum matter. *Advances in Physics: X*, 5(1):1797528, 2020.

[635] Juan Carrasquilla, Di Luo, Felipe Pérez, Ashley Milsted, Bryan K Clark, Maksims Volkovs, and Leandro Aolita. Probabilistic simulation of quantum circuits using a deep-learning architecture. *Physical Review A*, 104(3):032610, 2021.

[636] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2011. ISBN 9781107002173.

[637] Fionn D Malone, Robert M Parrish, Alicia R Welden, Thomas Fox, Matthias Degroote, Elica Kyoseva, Nikolaj Moll, Raffaele Santagati, and Michael Streif. Towards the simulation of large scale protein-ligand interactions on nisq-era quantum computers. *arXiv:2110.01589*, 2021.

[638] John M Shalf and Robert Leland. Computing beyond moore's law. *Computer*, 48(12):14–23, 2015.

[639] Ravi Athale and Demetri Psaltis. Optical computing: past and future. *Optics and Photonics News*, 27(6):32–39, 2016.

[640] Nasim Mohammadi Estakhri, Brian Edwards, and Nader Engheta. Inverse-designed metastructures that solve equations. *Science*, 363(6433):1333–1338, 2019.

[641] Gouhei Tanaka, Toshiyuki Yamane, Jean Benoit Héroux, Ryosho Nakane, Naoki Kanazawa, Seiji Takeda, Hidetoshi Numata, Daiju Nakano, and Akira Hirose. Recent advances in physical reservoir computing: A review. *Neural Networks*, 115:100–123, 2019.

[642] Tyler W Hughes, Ian AD Williamson, Momchil Minkov, and Shanhui Fan. Wave physics as an analog recurrent neural network. *Science advances*, 5(12):eaay6946, 2019.

[643] T Patrick Xiao, Christopher H Bennett, Ben Feinberg, Sapan Agarwal, and Matthew J Marinella. Analog architectures for neural network acceleration based on non-volatile memory. *Applied Physics Reviews*, 7(3):031301, 2020.

[644] Gordon Wetzstein, Aydogan Ozcan, Sylvain Gigan, Shanhui Fan, Dirk Englund, Marin Soljačić, Cornelia Denz, David AB Miller, and Demetri Psaltis. Inference in artificial intelligence with deep optics and photonics. *Nature*, 588(7836):39–47, 2020.

[645] Danijela Marković, Alice Mizrahi, Damien Querlioz, and Julie Grollier. Physics for neuromorphic computing. *Nature Reviews Physics*, 2(9):499–510, 2020.

[646] Logan G Wright, Tatsuhiro Onodera, Martin M Stein, Tianyu Wang, Darren T Schachter, Zoey Hu, and Peter L McMahon. Deep physical neural networks enabled by a backpropagation algorithm for arbitrary physical systems. *arXiv:2104.13386*, 2021.

[647] Genki Furuhata, Tomoaki Niiyama, and Satoshi Sunada. Physical deep learning based on optimal control of dynamical systems. *Physical Review Applied*, 15(3):034092, 2021.

[648] Xue Fan and Henry Markram. A brief history of simulation neuroscience. *Frontiers in Neuroinformatics*, 13, 2019.

[649] Henry Markram. The blue brain project. *Nature Reviews Neuroscience*, 7:153–160, 2006.

[650] Jeff Hawkins and Subutai Ahmad. Why neurons have thousands of synapses, a theory of sequence memory in neocortex. *Frontiers in Neural Circuits*, 10, 2016.

[651] David Soto, John Hodsoll, Pia Rotshtein, and Glyn W. Humphreys. Automatic guidance of attention from working memory. *Trends in Cognitive Sciences*, 12:342–348, 2008.

[652] André M. Bastos, W. Martin Usrey, Rick A Adams, George R. Mangun, Pascal Fries, and Karl J. Friston. Canonical microcircuits for predictive coding. *Neuron*, 76:695–711, 2012.

[653] Karl J. Friston. The free-energy principle: a unified brain theory? *Nature Reviews Neuroscience*, 11:127–138, 2010.

[654] Aboozar Taherkhani, Ammar Belatreche, Yuhua Li, Georgina Cosma, Liam P. Maguire, and T. Martin McGinnity. A review of learning in biologically plausible spiking neural networks. *Neural networks : the official journal of the International Neural Network Society*, 122:253–272, 2020.

[655] Tomer David Ullman, Noah D. Goodman, and Joshua B. Tenenbaum. Theory learning as stochastic search in the language of thought. *Cognitive Development*, 27:455–480, 2012.

[656] Danijela Marković, Alice Mizrahi, Damien Querlioz, and Julie Grollier. Physics for neuromorphic computing. *Nature Reviews Physics*, pages 1–12, 2020.

[657] Shimon Ullman. Using neuroscience to develop artificial intelligence. *Science*, 363:692 – 693, 2019.

[658] Dileep George, Miguel Lázaro-Gredilla, and J. Swaroop Guntupalli. From captcha to commonsense: How brain can teach us about artificial intelligence. *Frontiers in Computational Neuroscience*, 14, 2020.

[659] Demis Hassabis, Dharshan Kumaran, Christopher Summerfield, and Matthew M. Botvinick. Neuroscience-inspired artificial intelligence. *Neuron*, 95:245–258, 2017.

[660] Taco Cohen and Max Welling. Group equivariant convolutional networks. In *ICML*, 2016.

[661] Taco Cohen, Maurice Weiler, Berkay Kicanaoglu, and Max Welling. Gauge equivariant convolutional networks and the icosahedral cnn. In *ICML*, 2019.

[662] Victor Fung, Jiaxin Zhang, Eric Juarez, and Bobby G. Sumpter. Benchmarking graph neural networks for materials chemistry. *npj Computational Materials*, 7:1–8, 2021.

[663] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32:4–24, 2019.

[664] Pim de Haan, Taco Cohen, and Max Welling. Natural graph networks. *ArXiv*, abs/2007.08349, 2020.

[665] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *ArXiv*, abs/1810.00826, 2019.

[666] Haggai Maron, Heli Ben-Hamu, Nadav Shamir, and Yaron Lipman. Invariant and equivariant graph networks. *ArXiv*, abs/1812.09902, 2019.

[667] Dan Shiebler, Bruno Gavranovi'c, and Paul Wilson. Category theory in machine learning. *ArXiv*, abs/2106.07032, 2021.

[668] Jonathan M. Stokes, Kevin Yang, Kyle Swanson, Wengong Jin, Andres Cubillos-Ruiz, Nina M. Donghia, Craig R Macnair, Shawn French, Lindsey A. Carfrae, Zohar Bloom-Ackermann, Victoria M. Tran, Anush Chiappino-Pepe, Ahmed H. Badran, Ian W. Andrews, Emma J. Chory, George M. Church, Eric D. Brown, T. Jaakkola, Regina Barzilay, and James J. Collins. A deep learning approach to antibiotic discovery. *Cell*, 180:688–702.e13, 2020.

[669] Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur D. Szlam, and Pierre Vandergheynst. Geometric deep learning: Going beyond euclidean data. *IEEE Signal Processing Magazine*, 34:18–42, 2017.

[670] Gregor N. C. Simm, Robert Pinsler, and José Miguel Hernández-Lobato. Reinforcement learning for molecular design guided by quantum mechanics. In *ICML*, 2020.

[671] M. Zecevic, Devendra Singh Dhami, Petar Velickovic, and Kristian Kersting. Relating graph neural networks to structural causal models. *ArXiv*, abs/2109.04173, 2021.

[672] Jan E. Gerken, Jimmy Aronsson, Oscar Carlsson, Hampus Linander, Fredrik Ohlsson, Christoffer Petersson, and Daniel Persson. Geometric deep learning and equivariant neural networks. *ArXiv*, abs/2105.13926, 2021.

[673] Carlos Esteves. Theoretical aspects of group equivariant neural networks. *ArXiv*, abs/2004.05154, 2020.

[674] Jiri Blazek. Principles of solution of the governing equations. *Computational Fluid Dynamics: Principles and Applications, 3rd ed.; Blazek, J., Ed*, pages 121–166, 2015.

[675] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W. Battaglia. Learning mesh-based simulation with graph networks. *ArXiv*, abs/2010.03409, 2021.

[676] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter W. Battaglia. Learning to simulate complex physics with graph networks. *ArXiv*, abs/2002.09405, 2020.

[677] Erroll Wood, Tadas Baltrušaitis, Charlie Hewitt, Sebastian Dziadzio, Matthew Johnson, Virginia Estellers, Thomas J. Cashman, and Jamie Shotton. Fake it till you make it: Face analysis in the wild using synthetic data alone, 2021.

[678] Harkirat Singh Behl, Atilim Güneş Baydin, Ran Gal, Philip HS Torr, and Vibhav Vineet. Autosimulate:(quickly) learning synthetic data generation. In *European Conference on Computer Vision*, pages 255–271. Springer, 2020.

[679] Nikita Jaipuria, Xianling Zhang, Rohan Bhasin, Mayar Arafa, Punarjay Chakravarty, Shubham Shrivastava, Sagar Manglani, and Vidya N Murali. Deflating dataset bias using synthetic data augmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 772–773, 2020.

[680] Lukasz Romaszko, Christopher K. I. Williams, Pol Moreno, and Pushmeet Kohli. Vision-as-inverse-graphics: Obtaining a rich 3d explanation of a scene from a single image. *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, pages 940–948, 2017.

[681] Hiroharu Kato, Deniz Beker, Mihai Morariu, Takahiro Ando, Toru Matsuoka, Wadim Kehl, and Adrien Gaidon. Differentiable rendering: A survey. *ArXiv*, abs/2006.12057, 2020.

[682] Ivan Kobyzev, Simon Prince, and Marcus Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[683] Johann Brehmer and Kyle Cranmer. Flows for simultaneous manifold learning and density estimation. *ArXiv*, abs/2003.13913, 2020.

[684] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. *ArXiv*, abs/1605.08803, 2017.

[685] Hadi Salman, Payman Yadollahpour, Tom Fletcher, and Nematollah Batmanghelich. Deep diffeomorphic normalizing flows. *ArXiv*, abs/1810.03256, 2018.

[686] Will Grathwohl, Ricky T. Q. Chen, Jesse Bettencourt, Ilya Sutskever, and David Kristjanson Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *ArXiv*, abs/1810.01367, 2019.

[687] Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural spline flows. In *NeurIPS*, 2019.

[688] Isay Katsman, Aaron Lou, D. Lim, Qingxuan Jiang, Ser-Nam Lim, and Christopher De Sa. Equivariant manifold flows. *ArXiv*, abs/2107.08596, 2021.

[689] Jonas Köhler, Leon Klein, and Frank Noé. Equivariant flows: exact likelihood generative learning for symmetric densities. *ArXiv*, abs/2006.02425, 2020.

[690] Denis Boyda, Gurtej Kanwar, Sébastien Racanière, Danilo Jimenez Rezende, M. S. Albergo, Kyle Cranmer, Daniel C. Hackett, and Phiala Shanahan. Sampling using su(n) gauge equivariant flows. *ArXiv*, abs/2008.05456, 2021.

[691] Arleen Salles, Kathinka Evers, and Michele Farisco. Anthropomorphism in ai. *AJOB Neuroscience*, 11:88 – 95, 2020.

[692] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando GSL Brandao, David A Buell, et al. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, 2019.

[693] Kishor Bharti, Alba Cervera-Lierta, Thi Ha Kyaw, Tobias Haug, Sumner Alperin-Lea, Abhinav Anand, Matthias Degroote, Hermanni Heimonen, Jakob S Kottmann, Tim Menke, et al. Noisy intermediate-scale quantum (nisq) algorithms. *arXiv preprint arXiv:2101.08448*, 2021.

[694] Francesco Tacchino, Alessandro Chiesa, Stefano Carretta, and Dario Gerace. Quantum computers as universal quantum simulators: State-of-the-art and perspectives. *Advanced Quantum Technologies*, 3(3):1900052, 2020.

[695] Sam McArdle, Suguru Endo, Alán Aspuru-Guzik, Simon C Benjamin, and Xiao Yuan. Quantum computational chemistry. *Reviews of Modern Physics*, 92(1):015003, 2020.

[696] Marco Cerezo, Andrew Arrasmith, Ryan Babbush, Simon C Benjamin, Suguru Endo, Keisuke Fujii, Jarrod R McClean, Kosuke Mitarai, Xiao Yuan, Lukasz Cincio, et al. Variational quantum algorithms. *Nature Reviews Physics*, pages 1–20, 2021.

[697] Jacob Biamonte, Peter Wittek, Nicola Pancotti, Patrick Rebentrost, Nathan Wiebe, and Seth Lloyd. Quantum machine learning. *Nature*, 549(7671):195–202, 2017.

[698] Vedran Dunjko and Peter Wittek. A non-review of quantum machine learning: trends and explorations. *Quantum Views*, 4:32, 2020.

[699] Kosuke Mitarai, Makoto Negoro, Masahiro Kitagawa, and Keisuke Fujii. Quantum circuit learning. *Physical Review A*, 98(3): 032309, 2018.

[700] Ville Bergholm, Josh Izaac, Maria Schuld, Christian Gogolin, M Sohaib Alam, Shahnawaz Ahmed, Juan Miguel Arrazola, Carsten Blank, Alain Delgado, Soran Jahangiri, et al. Pennylane: Automatic differentiation of hybrid quantum-classical computations. *arXiv:1811.04968*, 2018.

[701] Michael Broughton, Guillaume Verdon, Trevor McCourt, Antonio J Martinez, Jae Hyeon Yoo, Sergei V Isakov, Philip Massey, Ramin Halavati, Murphy Yuezhen Niu, Alexander Zlokapa, et al. Tensorflow quantum: A software framework for quantum machine learning. *arXiv preprint arXiv:2003.02989*, 2020.

[702] D. Braben. Scientific freedom: The elixir of civilization. 2008.

[703] Eberhard O. Voit. Perspective: Dimensions of the scientific method. *PLoS Computational Biology*, 15, 2019.

[704] Thomas S. Kuhn and David Hawkins. The structure of scientific revolutions. *American Journal of Physics*, 31:554–555, 1962.

[705] Sauro Succi and Peter V. Coveney. Big data: the end of the scientific method? *Philosophical transactions. Series A, Mathematical, physical, and engineering sciences*, 377, 2019.

Acknowledgements

The authors would like to thank Tom Kalil and Adam Marblestone for their support, Joshua Elliot and Josh Tenenbaum for fruitful discussions, Sam Arbesman for useful reviews.

Competing interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to A.L.