The frontier of simulation-based inference

Kyle Cranmer[a,b,1], Johann Brehmer[a,b], and Gilles Louppe[c]

[a]Center for Cosmology and Particle Physics, New York University, New York, NY 10003; [b]Center for Data Science, New York University, New York, NY 10011; and [c]Montefiore Institute, University of Liège, B-4000 Liège, Belgium

Edited by Jitendra Malik, University of California, Berkeley, CA, and approved April 10, 2020 (received for review November 4, 2019)

Many domains of science have developed complex simulations to describe phenomena of interest. While these simulations provide high-fidelity models, they are poorly suited for inference and lead to challenging inverse problems. We review the rapidly developing field of simulation-based inference and identify the forces giving additional momentum to the field. Finally, we describe how the frontier is expanding so that a broad audience can appreciate the profound influence these developments may have on science.

statistical inference | implicit models | likelihood-free inference | approximate Bayesian computation | neural density estimation

Mechanistic models can be used to predict how systems will behave in a variety of circumstances. These run the gamut of distance scales, with notable examples including particle physics, molecular dynamics, protein folding, population genetics, neuroscience, epidemiology, economics, ecology, climate science, astrophysics, and cosmology. The expressiveness of programming languages facilitates the development of complex, high-fidelity simulations and the power of modern computing provides the ability to generate synthetic data from them. Unfortunately, these simulators are poorly suited for statistical inference. The source of the challenge is that the probability density (or likelihood) for a given observation—an essential ingredient for both frequentist and Bayesian inference methods—is typically intractable. Such models are often referred to as implicit models and contrasted against prescribed models where the likelihood for an observation can be explicitly calculated (1). The problem setting of statistical inference under intractable likelihoods has been dubbed likelihood-free inference—although it is a bit of a misnomer as typically one attempts to estimate the intractable likelihood, so we feel the term simulation-based inference is more apt.

The intractability of the likelihood is an obstruction for scientific progress as statistical inference is a key component of the scientific method. In areas where this obstruction has appeared, scientists have developed various ad hoc or field-specific methods to overcome it. In particular, two common traditional approaches rely on scientists to use their insight into the system to construct powerful summary statistics and then compare the observed data to the simulated data. In the first one, density estimation methods are used to approximate the distribution of the summary statistics from samples generated by the simulator (1). This approach was used for the discovery of the Higgs boson in a frequentist paradigm and is illustrated in Fig. 1*E*). Alternatively, a technique known as approximate Bayesian computation (ABC) (2, 3) compares the observed and simulated data based on some distance measure involving the summary statistics. ABC is widely used in population biology, computational neuroscience, and cosmology and is depicted in Fig. 1*A*. Both techniques have served a large and diverse segment of the scientific community.

Recently, the toolbox of simulation-based inference has experienced an accelerated expansion. Broadly speaking, three forces are giving new momentum to the field. First, there has been a significant cross-pollination between those studying simulation-based inference and those studying probabilistic models in machine learning (ML) (4), and the impressive growth of ML capabilities enables new approaches. Second, active learning—the idea of continuously using the acquired knowledge to guide the simulator—is being recognized as a key idea to improve the sample efficiency of various inference methods. A third direction of research has stopped treating the simulator as a black box and focused on integrations that allow the inference engine to tap into the internal details of the simulator directly.

Amidst this ongoing revolution, the landscape of simulation-based inference is changing rapidly. In this review we aim to provide the reader with a high-level overview of the basic ideas behind both old and new inference techniques. Rather than discussing the algorithms in technical detail, we focus on the current frontiers of research and comment on some ongoing developments that we deem particularly exciting.

Simulation-Based Inference

Simulators. Statistical inference is performed within the context of a statistical model, and in simulation-based inference the simulator itself defines the statistical model. For the purpose of this paper, a simulator is a computer program that takes as input a vector of parameters θ, samples a series of internal states or latent variables $z_i \sim p_i(z_i|\theta, z_{<i})$, and finally produces a data vector $x \sim p(x|\theta, z)$ as output. Programs that involve random samplings and are interpreted as statistical models are known as probabilistic programs, and simulators are an example. Within this general formulation, real-life simulators can vary substantially:

- The parameters θ describe the underlying mechanistic model and thus affect the transition probabilities $p_i(z_i|\theta, z_{<i})$. Typically the mechanistic model is interpretable by a domain scientist and θ has relatively few components and a fixed dimensionality. Examples include coefficients found in the Hamiltonian of a physical system, the virulence and incubation rate of a pathogen, or fundamental constants of Nature.
- The latent variables z that appear in the data-generating process may directly or indirectly correspond to a physically meaningful state of a system, but typically this state is unobservable in practice. The structure of the latent space varies substantially between simulators. The latent variables may be continuous or discrete and the dimensionality of the latent space may be fixed or may vary, depending on the control flow of the simulator. The simulation can freely combine deterministic and stochastic steps. The deterministic components of the simulator may be differentiable or may involve discontinuous control flow elements. In practice, some simulators may provide convenient access to the latent variables, while others are effectively

This paper results from the Arthur M. Sackler Colloquium of the National Academy of Sciences, "The Science of Deep Learning," held March 13–14, 2019, at the National Academy of Sciences in Washington, DC. NAS colloquia began in 1991 and have been published in PNAS since 1995. From February 2001 through May 2019 colloquia were supported by a generous gift from The Dame Jillian and Dr. Arthur M. Sackler Foundation for the Arts, Sciences, & Humanities, in memory of Dame Sackler's husband, Arthur M. Sackler. The complete program and video recordings of most presentations are available on the NAS website at http://www.nasonline.org/science-of-deep-learning.

Author contributions: K.C., J.B., and G.L. performed research and wrote the paper.

The authors declare no competing interest.

This article is a PNAS Direct Submission.

[1]To whom correspondence may be addressed. Email: kyle.cranmer@nyu.edu.

First published May 29, 2020.

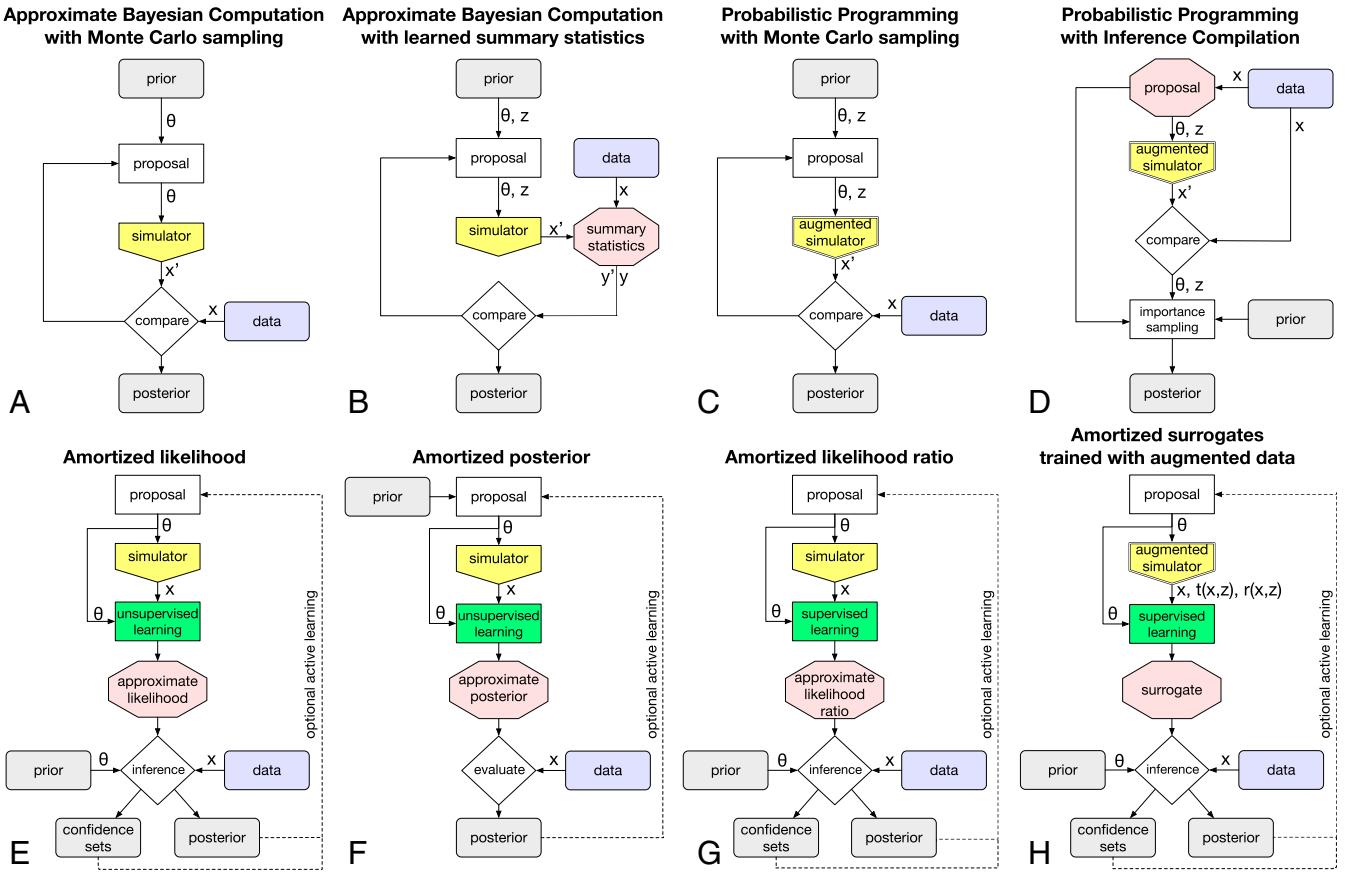

Fig. 1. (*A–H*) Overview of different approaches to simulation-based inference.

black boxes. Any given simulator may combine these different aspects in almost any way.

- Finally, the output data x correspond to the observations. They can range from a few unstructured numbers to high-dimensional and highly structured data, such as images or geospatial information.

For instance, particle physics processes often depend only on a small number of parameters of interest such as particle masses or coupling strengths. The latent process combines a high-energy interaction, rigorously described by a quantum-field theory, with the passage of the resulting particles through an incredibly complex detector, most accurately modeled with stochastic simulations with billions of latent variables. Epidemiological simulations can be based on a network structure with geospatial properties, and the latent process consists of many repeated structurally identical stochastic time steps. In contrast, cosmological simulations of the evolution of the Universe may consist of a highly structured stochastic initial state followed by a smooth, deterministic time evolution.

These differences lead to an absence of a one-size-fits-all inference method. In this review we aim to clarify the considerations needed to choose the most appropriate approach for a given problem.

Inference. Scientific inference tasks differ by what is being inferred: Given observed data x, is the goal to infer the input parameters θ, or the latent variables z, or both? We will focus on the common problem of inferring θ in a parametric setting, we will comment on methods that allow inference on z, and we will not focus on nonparametric inverse problems. Sometimes only a subset of the parameters is of interest, while the rest are nuisance parameters.

Inference may be performed either in a frequentist or in a Bayesian approach and may be limited to point estimates $\hat{\theta}(x)$ or extended to include a probabilistic notion of uncertainty. In the frequentist case, confidence sets are often formed from inverting hypothesis tests based on the likelihood-ratio test statistic. In Bayesian inference, the goal is typically to calculate the posterior $p(\theta|x) = p(x|\theta)p(\theta)/\int \mathrm{d}\theta' p(x|\theta')p(\theta')$ for observed data x and a given prior $p(\theta)$. In both cases the likelihood function $p(x|\theta)$ is a key ingredient.

The fundamental challenge for simulation-based inference problems is that the likelihood function $p(x|\theta)$ implicitly defined by the simulator is typically not tractable, as it corresponds to an integral over all possible trajectories through the latent space (i.e., all possible execution traces of the simulator). That is,

$$p(x|\theta) = \int \mathrm{d}z\, p(x, z|\theta), \qquad [1]$$

where $p(x, z|\theta)$ is the joint probability density of data x and latent variables z. For real-life simulators with large latent spaces, it is clearly impossible to compute this integral explicitly. Since the likelihood function is the central ingredient in both frequentist and Bayesian inference, this is a major challenge for inference in many fields. This paper reviews simulation-based or likelihood-free inference techniques that enable frequentist or Bayesian inference despite this intractability. These methods can be seen as a specialization of inverse uncertainty quantification (UQ) on the model parameters in situations with accurate, stochastic simulators.

In practice, an important distinction is that between inference based on a single observation and that based on multiple independent and identically distributed (i.i.d.) observations. In the second case, the likelihood factorizes into individual likelihood terms for each i.i.d. observation, as $p(x|\theta) = \prod_i p_{\text{individual}}(x_i|\theta)$. For example, time-series data are typically non-i.i.d. and must be treated as a single high-dimensional observation, whereas the analysis of collision data in the search for the Higgs boson constitutes a dataset with many i.i.d. measurements. This distinction is important when it comes to the computational cost of an inference technique, as inference in the i.i.d. case will necessitate many repeated evaluations of the individual likelihood $p_{\text{individual}}(x_i|\theta)$.

Traditional Methods. The problem of inference without tractable likelihoods is not a new one, and two major approaches have been developed to address it. Arguably the most well known is ABC (2, 3). Until recently, it was so established that the terms "likelihood-free inference" and "ABC" were often used interchangeably. In the simplest form of rejection ABC, the parameters θ are drawn from the prior, the simulator is run with those values to sample $x_{\text{sim}} \sim p(\cdot|\theta)$, and θ is retained as the posterior sample if the simulated data are sufficiently close to the observed data. In essence, the likelihood is approximated by the probability that the condition $\rho(x_{\text{sim}}, x_{\text{obs}}) < \epsilon$ is satisfied, where ρ is some distance measure and ϵ is a tolerance. The accepted samples then follow an approximate version of the posterior. We show a schematic workflow of this algorithm in Fig. 1*A* (for a more elaborate Markov chain Monte Carlo algorithm with a proposal function).

In the limit $\epsilon \to 0$, inference with ABC becomes exact, but for continuous data the acceptance probability vanishes. In practice, small values of ϵ require unfeasibly many simulations. For large ϵ, sample efficiency is increased at the expense of inference quality. Similarly, the sample efficiency of ABC scales poorly to high-dimensional data x. Since the data immediately affect the rejection process (and in more advanced ABC algorithms the proposal distribution), inference for new observations requires repeating the entire inference algorithm. ABC is thus best suited for the case of a single observation or at most a few i.i.d. data points. Lacking space to do the vast ABC literature justice, we refer the reader to a review of ABC methods, see ref. 5, and highlight the combination with Markov chain Monte Carlo (MCMC) (6) and sequential Monte Carlo (SMC) (7, 8).

The second classical approach to simulation-based inference is based on creating a model for the likelihood by estimating the distribution of simulated data with histograms or kernel density estimation (1). Frequentist and Bayesian inference then proceeds as if the likelihood were tractable. We sketch this algorithm in Fig. 1*E* (replacing the green learning step with a classical density estimation method). This approach has enough similarities to ABC to be dubbed "approximate frequentist computation" by the authors of ref. 9. One advantage over ABC is that it is amortized: After an upfront computational cost at the simulation and density estimation stage, new data points can be evaluated efficiently. In Fig. 1*E* this is depicted by the blue "data" box entering only at the inference stage and not affecting the simulation step. This property makes density estimation-based inference particularly well suited for problems with many i.i.d. observations, a key reason for its widespread use in particle physics measurements.

Both of the traditional approaches suffer from the curse of dimensionality: In the worst case, the required number of simulations increases exponentially with the dimension of the data x. Therefore both approaches rely on low-dimensional summary statistics $y(x)$ and the quality of inference is tied to how well those summaries retain information about the parameters θ. Traditionally, the development of powerful summary statistics has been the task of a domain expert, and the summary statistics have been prescribed prior to inference.

Frontiers of Simulation-Based Inference. These traditional simulation-based inference techniques have played a key role in several fields for years. However, they suffer from shortcomings in three crucial aspects:

- Sample efficiency: The number of simulated samples needed to provide a good estimate of the likelihood or posterior can be prohibitively expensive.
- Quality of inference: The reduction of the data to low-dimensional summary statistics invariably discards some of the information in the data about θ, which results in a loss in statistical power. Large values of the ϵ parameter in ABC or the bandwidth parameter for kernel density estimation lead to poor approximations of the true likelihood. Both reduce the overall quality of inference.
- Amortization: Performing inference with ABC for a new set of observed data requires repeating most steps of the inference chain, in particular if the proposal distribution depends on the observed data. The method scales poorly when applied to large numbers of observations. On the other hand, inference based on density estimation is amortized: The computationally expensive steps do not have to repeated for new observations. This is particularly desirable for the case with i.i.d. observations.

In recent years, new capabilities have become available that let us improve all three of these aspects. We loosely group them into three main directions of progress:

1) The ML revolution allows us to work with higher-dimensional data, which can improve the quality of inference. Inference methods based on neural network surrogates are directly benefiting from the impressive rate of progress in deep learning.
2) Active learning methods can systematically improve sample efficiency, letting us tackle more computationally expensive simulators.
3) The deep integration of automatic differentiation and probabilistic programming into the simulation code, as well as the augmentation of training data with information that can be extracted from the simulator, is changing the way the simulator is treated in inference: It is no longer a black box, but exposed to the inference workflow.

A Revolution in Machine Learning. Over the last decade, ML techniques, in particular deep neural networks, have turned into versatile, powerful, and popular tools for a variety of problems (10). Neural networks initially demonstrated breakthroughs in supervised learning tasks such as classification and regression. They can easily be composed to solve higher-level tasks, lending themselves to problems with a hierarchical or compositional structure. Architectures tailored to various data structures have been developed, including dense or fully connected networks aimed at unstructured data, convolutional neural networks that leverage spatial structure for instance in image data, recurrent neural networks for variable-length sequences, and graph neural networks for graph-structured data. Choosing an architecture well suited for a specific data structure is an example of inductive bias, which more generally refers to the assumptions inherent in a learning algorithm independent of the data. Inductive bias is one of the key ingredients behind most successful applications of deep learning, although it is difficult to characterize its role precisely.

One area where neural networks are being actively developed is density estimation in high dimensions: Given a set of points $\{x\} \sim p(x)$, the goal is to estimate the probability density $p(x)$.

As there are no explicit labels, this is usually considered an unsupervised learning task. We have already discussed that classical methods based for instance on histograms or kernel density estimation do not scale well to high-dimensional data. In this regime, density estimation techniques based on neural networks are becoming more and more popular. One class of these neural density estimation techniques is normalizing flows (11–26), in which variables described by a simple base distribution $p(u)$ such as a multivariate Gaussian are transformed through a parameterized invertible transformation $x = g_\phi(u)$ that has a tractable Jacobian. The target density $p_g(x)$ is then given by the change-of-variables formula as a product of the base density and the determinant of the transformation's Jacobian. Several such steps can be stacked, with the probability density "flowing" through the successive variable transformations. The parameters ϕ of the transformations are trained by maximizing the likelihood of the observed data under the model $p_g(x_{\text{obs}})$, resulting in a model density that approximates the true, unknown density $p(x)$. In addition to having a tractable density, it is possible to generate data from the model by drawing the hidden variables u from the base distribution and applying the flow transformations. Neural density estimators have been generalized to model the dependency on additional inputs, i.e., to model a conditional density such as the likelihood $p(x|\theta)$ or posterior $p(\theta|x)$.

Generative adversarial networks (GANs) are an alternative type of generative model based on neural networks. Unlike in normalizing flows, the transformation implemented by the generator is not restricted to be invertible. While this allows for more expressiveness, the density defined by the generator is intractable. Since maximum likelihood is not a possible training objective, the generator is pitted against an adversary, whose role is to distinguish the generated data from the target distribution. We will later discuss how the same idea can be used for simulation-based inference, using an idea known as the "likelihood-ratio trick."

Active Learning. A simple, but very impactful idea is to run the simulator at parameter points θ that are expected to increase our knowledge the most. This can be done iteratively such that after each simulation the knowledge resulting from all previous runs is used to guide which parameter point should be used next. There are multiple technical realizations of this idea of active learning. It is commonly applied in a Bayesian setting, where the posterior can be continuously updated and used to steer the proposal distribution of simulator parameters (27–33). But it applies equally well to efficiently calculating frequentist confidence sets (34–36). Even simple implementations can lead to a substantial improvement in sample efficiency.

Similar ideas are discussed in the context of decision making, experimental design, and reinforcement learning, and we expect further improvements in inference algorithms from the cross-pollination between these fields. For instance, a question that is occasionally discussed in the context of reinforcement learning (37, 38) or Bayesian optimization (39), but has not yet been applied to the likelihood-free setting, is how to make use of multifidelity simulators offering multiple levels of precision or approximations.

Integration and Augmentation. Both ML and active learning can substantially improve quality of inference and sample efficiency compared to classical methods. However, overall they do not change the basic approach to simulation-based inference dramatically: They still treat the simulator as a generative black box that takes parameters as input and provides data as output, with a clear separation between the simulator and the inference engine. A third direction of research is changing this perspective, by opening the black box and integrating inference and simulation more tightly.

One example of this shift is the probabilistic programming paradigm. Gordon et al. (40) describe probabilistic programs as the usual functional or imperative programs with two added constructs: 1) the ability to draw values at random from distributions and 2) the ability to condition values of variables in a program via observations. We have already described simulators as probabilistic programs focusing on the first construct, which does not require opening the black box. However, conditioning on the observations requires a deeper integration as it involves controlling the randomness in the generative process. This approach abstracts the capabilities needed to implement particle filters and SMC (41). Previously, this required writing the program in a special-purpose language; however, recent work allows these capabilities to be added to existing simulators with minimal changes to their codebase (42). Ultimately, probabilistic programming aims at providing the tools to infer the incredibly complex space of all execution traces of the simulator conditioned on the observation.

A complementary development is the observation that additional information that characterizes the latent data-generating process can be extracted from the simulator and used to augment the data used to train surrogates. Those developing inference algorithms and those familiar with the details of the simulator should consider whether, in addition to the sole ability to sample $x \sim p(x|\theta)$, the following quantities are well defined and tractable:

1) $p(x|z,\theta)$
2) $t(x,z|\theta) \equiv \nabla_\theta \log p(x,z|\theta)$
3) $\nabla_z \log p(x,z|\theta)$
4) $r(x,z|\theta,\theta') \equiv p(x,z|\theta)/p(x,z|\theta')$
5) $\nabla_\theta(x,z)$
6) $\nabla_z x$.

These quantities can then be used to augment the usual output x from the simulator and can be exploited in supervised learning objectives and can dramatically increase the sample efficiency for surrogate training (43–45), as we will detail later.

Many of the quantities above involve derivatives, which can now be efficiently calculated using automatic differentiation (often referred to simply as autodiff) (46). Autodiff is a family of techniques similar to but more general than the backpropagation algorithm that is ubiquitous in deep learning. Automatic differentiation, like probabilistic programming, involves nonstandard interpretations of the simulation code and has been developed by a small but established field of computer science. In recent years several researchers have advocated that deep learning would be better described as differential programming (47, 48). With this view, incorporating autodiff into existing simulation codes is a more direct way to exploit the advances in deep learning than trying to incorporate domain knowledge into an entirely foreign substrate such as a deep neural network.

Extracting the necessary information from the simulator again requires integration deep in the code. While technologies to incorporate the probabilistic programming paradigm into existing code bases are just emerging, the development of tools to enable autodiff in the most commonly used scientific programming languages is well advanced. We highlight that two of the quantities listed above (quantities 2 and 3) involve both autodiff and probabilistic programming. The integration of inference and simulation as well as the idea of augmenting the training data with additional quantities has the potential to change the way we think about simulation-based inference. In particular, this perspective can influence the way simulation codes are developed to provide these new capabilities.

Workflows for Simulation-Based Inference

This wide array of capabilities can be combined in different inference workflows. As a guideline through this array of different

workflows, let us first discuss common building blocks and the different approaches that can be taken in each of these components. In Fig. 1 and the following sections we will then piece these blocks together into different inference algorithms.

An integral part of all inference methods is running the simulator, visualized as a yellow pentagon in Fig. 1. The parameters at which the simulator is run are drawn from some proposal distribution, which may or may not depend on the prior in a Bayesian setting, and can be chosen either statically or iteratively with an active learning method. Next, the potentially high-dimensional output from the simulator may be used directly as input to the inference method or reduced to low-dimensional summary statistics.

The inference techniques can be broadly separated into those which, like ABC, use the simulator itself during inference and methods which construct a surrogate model and use that for inference. In the first case, the output of the simulator is directly compared to data (Fig. 1 *A–D*). In the latter case, the output of the simulator is used as training data for an estimation or ML stage, shown as the green boxes in Fig. 1 *E–H*. The resulting surrogate models, shown as red hexagons, are then used for inference.

The algorithms address the intractability of the true likelihood in different ways: Some methods construct a tractable surrogate for the likelihood function and others that for the likelihood-ratio function, both of which make frequentist inference straightforward. In other methods, the likelihood function never appears explicitly, for instance when it is implicitly replaced by rejection probability.

The final target for Bayesian inference is the posterior. Methods differ in whether they provide access to samples of parameter points sampled from the posterior, for instance from MCMC or ABC, or a tractable function that approximates the posterior function. Similarly, some methods require specifying which quantities are to be inferred early on in the workflow, while others allow this decision to be postponed.

Using the Simulator Directly during Inference. Let us now discuss how these blocks and computational capabilities can be combined into inference techniques, beginning with those which, like ABC, use the simulator directly during inference. We sketch some of these algorithms in Fig. 1 *A–D*.

A reason for the poor sample efficiency of the original rejection ABC algorithm is that the simulator is run at parameter points drawn from the prior, which may have a large mass in regions that are in strong disagreement with the observed data. Different algorithms have been proposed that instead run the simulator at parameter points that are expected to improve the knowledge on the posterior the most (27–31). Compared to vanilla ABC, these techniques improve sample efficiency, although they still require the choice of summary statistics, distance measure ρ, and tolerance ϵ.

In the case where the final stage of the simulator is tractable or the simulator is differentiable (respectively, properties 1 and 6 from the list in *Integration and Augmentation*), asymptotically exact Bayesian inference is possible (43) without relying on a distance tolerance or summary statistics, removing ABC's main limitations in terms of quality of inference.

The probabilistic programming paradigm presents a more fundamental change to how inference is performed. First, it requires the simulator to be written in a probabilistic programming language, although recent work allows these capabilities to be added to existing simulators with minimal changes to their codebase (42). In addition, probabilistic programming requires either a tractable likelihood for the final step $p(x|z,\theta)$ (quantity 1) or the introduction of an ABC-like comparison. When these criteria are satisfied, several inference algorithms exist that can draw samples from the posterior $p(\theta, z|x)$ of the input parameters θ and the latent variables z given some observed data x. These techniques are based either on MCMC (Fig. 1*C*) or on training a neural network to provide proposal distributions (49) as shown in Fig. 1*D*. The key difference from ABC is that the inference engine controls all steps in the program execution and can bias each draw of random latent variables to make the simulation more likely to match the observed data, improving sample efficiency.

A strength of these algorithms is that they allow us to infer not only the input parameters into the simulator, but also the entire latent process leading to a particular observation. This allows us to answer entirely different questions about scientific processes, adding a particular kind of physical interpretability that methods based on surrogates do not possess. While standard ABC algorithms in principle allow for inference on z, probabilistic programming solves this task more efficiently.

Surrogate Models. A key disadvantage of using the simulator directly during inference is the lack of amortization. When new observed data become available, the whole inference chain has to be repeated. By training a tractable surrogate or emulator for the simulator, inference is amortized: After an upfront simulation and training phase, new data can be evaluated very efficiently. This approach scales particularly well to data consisting of many i.i.d. observations. As discussed in *Traditional Methods*, this is not a new idea, and well-established methods use classical density estimation techniques to create a surrogate model for the likelihood function. But the new computational capabilities discussed in *Simulation-Based Inference* have given new momentum to this class of inference techniques, some of which are visualized in Fig. 1 *E–H*.

A powerful probabilistic approach is to train neural conditional density estimators such as normalizing flows as a surrogate for the simulator. The conditional density can be defined in two directions: The network can learn either the posterior $p(\theta|x)$ (32, 50–54) or the likelihood $p(x|\theta)$ (33, 55–57). We show these three techniques in Fig. 1 *E* and *F*; note that the likelihood surrogate algorithm is structurally identical to the classical density estimation-based approach, but uses more powerful density estimation techniques.

Relatedly, neural networks can be trained to learn the likelihood-ratio function $p(x|\theta_0)/p(x|\theta_1)$ or $p(x|\theta_0)/p(x)$, where in the latter case the denominator is given by a marginal model integrated over a proposal or the prior (4, 58–65). We sketch this approach in Fig. 1*G*. The key idea is closely related to the discriminator network in GANs mentioned above: A classifier is trained using supervised learning to discriminate two sets of data, although in this case both sets come from the simulator and are generated for different parameter points θ_0 and θ_1. The classifier output function can be converted into an approximation of the likelihood ratio between θ_0 and θ_1! This manifestation of the Neyman–Pearson lemma in an ML setting is often called the likelihood-ratio trick.

These three surrogate-based approaches are all amortized: After an upfront simulation and training phase, the surrogates can be evaluated efficiently for arbitrary data and parameter points. They require an upfront specification of the parameters of interest, and the network then implicitly marginalizes over all other (latent) variables in the simulator. All three classes of algorithms can employ active learning elements such as an iteratively updated proposal distribution to guide the simulator parameters θ toward the relevant parameter region, improving sample efficiency. Using neural networks eliminates the requirement of low-dimensional summary statistics, leaving it to the employed model to learn the structures in high-dimensional data and potentially improving quality of inference.

Despite these fundamental similarities, there are some differences between emulating the likelihood, the likelihood ratio, and the posterior. Learning the posterior directly provides the

main target quantity in Bayesian inference but induces a prior dependence at every stage of the inference method. Learning the likelihood or the likelihood ratio enables frequentist inference or model comparisons, although for Bayesian inference an additional MCMC or quantity 6 step is necessary to generate samples from the posterior. The prior independence of likelihood or likelihood-ratio estimators also leads to extra flexibility to change the prior during inference. An advantage of training a generative model to approximate the likelihood or posterior over learning the likelihood-ratio function is the added functionality of being able to sample from the surrogate model. On the other hand, learning the likelihood or posterior is an unsupervised learning problem, whereas estimating the likelihood ratio through a classifier is an example of supervised learning and often a simpler task. Since for the higher-level inference goal the likelihood and the likelihood ratio can be used interchangeably, learning a surrogate for the likelihood-ratio function may often be more efficient.

Another strategy that allows us to leverage supervised learning is based on extracting additional quantities from the simulator that characterize the likelihood of the latent process (e.g., quantities 2 and 4 from the list in *Integration and Augmentation*). This additional information can be used to augment the training data for surrogate models. The resulting supervised learning task can often be solved more efficiently, ultimately improving the sample efficiency in the inference task (9, 44, 45, 66).

Surrogate-based approaches benefit from imposing suitable inductive bias for a given problem. It is widely acknowledged that the network architecture of a neural surrogate should be chosen according to the data structure (e.g., images, sequences, or graphs). Another, potentially more consequential, way of imposing inductive bias is to have the surrogate model reflect the causal structure of the simulator. Manually identifying the relevant structures and designing appropriate surrogate architectures is very domain specific, although has been shown to improve the performance on some problems (67–69). Recently attempts have been made to automate the process of creating surrogates that mimic the simulation (70). Looking farther ahead, we would like to learn surrogates that reflect the causal structure of a coarse-grained system. If this is possible, it would allow the surrogate to model only the relevant degrees of freedom for the phenomena that emerge from the underlying mechanistic model.

Preprocessing and Postprocessing. There are a number of additional steps that can surround these core inference algorithms, either in the form of preprocessing steps that precede the main inference stage or as an afterburner following the main inference step.

One preprocessing step is to learn powerful summaries $y(x)$. Because of the curse of dimensionality, both ABC and classical density estimation-based inference methods require a compression of the data into low-dimensional summary statistics. They are usually prescribed (i.e., hand chosen by domain scientists based on their intuition and knowledge of the problem at hand), but the resulting summaries will generally lose some information compared to the original data. A minimally invasive extension of these algorithms is to first learn summary statistics that have certain optimality properties, before running a standard inference algorithm such as ABC. We sketch this approach in Fig. 1*B* for ABC, but it applies equally to inference based on density estimation.

The score $t(x|\theta) \equiv \nabla_\theta p(x|\theta)$, the gradient of the log (marginal) likelihood with respect to the parameters of interest, defines such a vector of optimal summary statistics: In a neighborhood of θ, the score components are sufficient statistics, and they can be used for inference without loss of information. Just like the likelihood, the score itself is generally intractable, but it can be estimated based on quantity 5 and an exponential family approximation (71, 72). If quantity 2 is available, augmented data extracted from the simulator can instead be used to train a neural network to estimate the score (44) without requiring such an approximation. Learned summaries can also be made robust with respect to nuisance parameters (73, 74).

Even if it is not necessary to reduce the data to low-dimensional summary statistics, in some fields the measured raw or "low-level" data can be very high dimensional. It is then common practice to compress them to a more manageable set of "high-level" features of moderate dimensionality and to use these compressed data as input to the inference workflow.

Inference compilation (49) is a preprocessing step for probabilistic programming algorithms, shown in Fig. 1*D*. Initial runs of the simulator are used to train a neural network used for sequential importance sampling of both θ and z.

After the completion of the core inference workflow, an important question is whether the results are reliable: Can the outcome be trusted in the presence of imperfections such as limited sample size, insufficient network capacity, or inefficient optimization?

One solution is to calibrate the inference results. Using the ability of the simulator to generate data for any parameter point, we can use a parametric bootstrap approach to calculate the distribution of any quantity involved in the inference workflow. These distributions can be used to calibrate the inference procedure to provide confidence sets and posteriors with proper coverage and credibility (9, 60). While possible in principle, such procedures may require a large number of simulations. Other diagnostic tools that can be applied at the end of the inference stage involve training classifiers to distinguish data from the surrogate model and the true simulator (60), self-consistency checks (9, 60), ensemble methods, and comparing distributions of network output against known asymptotic properties (75–77). Some of these methods may be used for uncertainty estimation, although the statistical interpretation of such error bars is not always obvious.

None of these diagnostics address the issues encountered if the model is misspecified and the simulator is not an accurate description of the system being studied. Model misspecification is a problem that plagues inference with both prescribed and implicit models equally. Usually this is addressed by expanding the model to have more flexibility and introducing additional nuisance parameters.

Recommendations. The considerations needed to choose which of the approaches described above is best for a given problem will include the inference goals, the dimensionality of model parameters, the latent variables, and the data; whether good summary statistics are available; the internal structure of the simulator; the computational cost of the simulator; the level of control over how the simulator is run; and whether the simulator is a black box or whether any of the quantities discussed in *Integration and Augmentation* can be extracted from it. Nevertheless, we believe that the existing body of research lets us provide a few general guidelines.

First, if any of the quantities discussed in *Frontiers of Simulation-Based Inference* are available, they should be leveraged. Powerful algorithms are available for the case with differentiable simulators (43), for simulators for which the joint likelihood of data and latent variables is accessible (44), and for simulators explicitly written as a probabilistic model in a probabilistic programming framework (78). Probabilistic programming is also the most versatile approach when the goal is not only inference on the parameters θ, but also inference on the latent variables z.

If powerful low-dimensional summary statistics are established, traditional techniques can still offer a reasonable performance. In most cases, however, we recommend trying methods based on training a neural network surrogate for the likelihood

(33, 56) or the likelihood ratio (60, 62, 64). If generating synthetic data from the surrogate is not important, learning the likelihood ratio rather than the likelihood allows us to leverage powerful supervised learning methods.

Finally, active learning techniques can improve the sample efficiency for all inference techniques. There is a tradeoff between active learning, which tailors the efficiency to a particular observed dataset, and amortization, which benefits from surrogates that are agnostic about the observed data. A good compromise here will depend on the number of observations and the sharpness of the posterior compared to the prior.

Discussion

Until recently, scientists confronted with inverse problems and a complex simulator as a forward model had little recourse other than to choose ABC or methods based on classical density estimation techniques. While these approaches have served some domains of science quite well, they have relied heavily on the labor-intensive process of experts providing powerful summary statistics. As a result, there is a frontier beyond which these traditional methods are less useful and new techniques are needed.

The term likelihood-free inference has served as a point of convergence for what were previously disparate communities, and a new lingua franca has emerged. This has catalyzed significant cross-pollination and led to a renaissance in simulation-based inference. The advent of powerful ML methods is enabling practitioners to work directly with high-dimensional data and to reduce the reliance on expert-crafted summary statistics. New programming paradigms such as probabilistic programming and differentiable programming provide new capabilities that enable entirely new approaches to simulation-based inference. Finally, taking a more systems-level view of simulation-based inference that brings together the statistical and computational considerations has taken root. Here active learning is leading the way, but we expect more advances like this as simulation-based inference matures.

The rapidly advancing frontier means 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.

Data Availability. There are no data associated with this paper.

ACKNOWLEDGMENTS. K.C. and J.B. are supported by the National Science Foundation under Awards ACI-1450310, OAC-1836650, and OAC-1841471 and by the Moore–Sloan data science environment at New York University. G.L. is the recipient of the University of Liège–Network Research Belgium (NRB) Chair on Big Data and is thankful for the support of NRB.

1. P. J. Diggle, R. J. Gratton, Monte Carlo methods of inference for implicit statistical models. *J. R. Stat. Soc. Ser. B* **46**, 193–212 (1984).
2. D. B. Rubin, Bayesianly justifiable and relevant frequency calculations for the applied statistician. *Ann. Stat.* **12**, 1151–1172 (1984).
3. M. A. Beaumont, W. Zhang, D. J. Balding, Approximate Bayesian computation in population genetics. *Genetics* **162**, 2025–2035 (2002).
4. S. Mohamed, B. Lakshminarayanan, Learning in implicit generative models. arXiv:1610.03483 (11 October 2016).
5. S. A. Sisson, Y. Fan, M. Beaumont, *Handbook of Approximate Bayesian Computation* (Chapman and Hall/CRC, 2018).
6. P. Marjoram, J. Molitor, V. Plagnol, S. Tavaré, Markov chain Monte Carlo without likelihoods. *Proc. Natl. Acad. Sci. U.S.A.* **100**, 15324–15328 (2003).
7. S. A. Sisson, Y. Fan, M. M. Tanaka, Sequential Monte Carlo without likelihoods. *Proc. Natl. Acad. Sci. U.S.A.* **104**, 1760–1765 (2007).
8. G. W. Peters, Y. Fan, S. A. Sisson, On sequential Monte Carlo, partial rejection control and approximate Bayesian computation. *Stat. Comput.* **22**, 1209–1222 (2012).
9. J. Brehmer, K. Cranmer, G. Louppe, J. Pavez, A guide to constraining effective field theories with machine learning. *Phys. Rev. D* **98**, 052004 (2018).
10. Y. LeCun, Y. Bengio, G. Hinton, Deep learning. *Nature* **521**, 436–444 (2015).
11. L. Dinh, D. Krueger, Y. Bengio, NICE: Non-linear independent components estimation. arXiv:1410.8516 (30 October 2014).
12. L. Dinh, J. Sohl-Dickstein, S. Bengio, "Density estimation using real NVP" in *5th International Conference on Learning Representations, ICLR 2017. April 24-26, 2017, Conference Track Proceedings* (Toulon, France, 2017).
13. D. P. Kingma, P. Dhariwal, "Glow: Generative flow with invertible 1×1 convolutions" in *Advances in Neural Information Processing Systems 2018* (2018), pp. 10215–10224.
14. M. Germain, K. Gregor, I. Murray, H. Larochelle, "MADE: Masked autoencoder for distribution estimation" in *32nd International Conference on Machine Learning, ICML 2015,* vol. 2, pp. 881–889.
15. B. Uria, M. A. Côté, K. Gregor, I. Murray, H. Larochelle, Neural autoregressive distribution estimation. *J. Mach. Learn. Res.* **17**, 7184–7220 (2016).
16. A. Van Den Oord, N. Kalchbrenner, K. Kavukcuoglu, "Pixel recurrent neural networks" in *33rd International Conference on Machine Learning, ICML 2016* (2016), vol. 4, pp. 2611–2620.
17. A. Van Den Oord *et al.*, "Conditional image generation with PixelCNN decoders" in *Advances in Neural Information Processing Systems* (2016), pp. 4797–4805.
18. A. van den Oord *et al.*, WaveNet: A generative model for raw audio. arXiv:1609.03499 (12 September 2016).
19. D. P. Kingma *et al.*, Improving variational inference with inverse autoregressive flow. arXiv:1606.04934 (15 June 2016).
20. G. Papamakarios, T. Pavlakou, I. Murray, "Masked autoregressive flow for density estimation" in *Advances in Neural Information Processing Systems 2017-December* (2017), pp. 2339–2348.
21. C. W. Huang, D. Krueger, A. Lacoste, A. Courville, "Neural autoregressive flows" in *35th International Conference on Machine Learning, ICML 2018* (2018), vol. 5, pp. 3309–3324.
22. A. Wehenkel, G. Louppe, Unconstrained monotonic neural networks. arXiv:1908.05164 (14 August 2019).
23. C. Durkan, A. Bekasov, I. Murray, G. Papamakarios, Cubic-spline flows. arXiv:1906.02145 (5 June 2019).
24. C. Durkan, A. Bekasov, I. Murray., G. Papamakarios, Neural spline flows. arXiv:1906.04032 (10 June 2019).
25. M. A. Hjortsø, P. Wolenski, "Neural ordinary differential equations" in *Linear Mathematical Models in Chemical Engineering abs/1806* (2018), pp. 123–145.
26. W. Grathwohl, R. T. Q. Chen, J. Bettencourt, D. Duvenaud, Scalable reversible generative models with free-form continuous dynamics" in *International Conference on Learning Representations* (2019).
27. E. Meeds, M. Welling, "Gps-abc: Gaussian process surrogate approximate Bayesian computation" in *Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence, UAI'14* (AUAI Press, Arlington, VA, 2014), pp. 593–602.
28. M. U. Gutmann, J. Corander, Bayesian optimization for likelihood-free inference of simulator-based statistical models. *J. Mach. Learn. Res.* **17**, 4256–4302 (2016).
29. E. Meeds, M. Welling, "Optimization Monte Carlo: Efficient and embarrassingly parallel likelihood-free inference" in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2, NIPS'15* (MIT Press, Cambridge, MA, 2015), pp. 2080–2088.
30. M. Järvenpää, M. U. Gutmann, A. Pleska, A. Vehtari, P. Marttinen, Efficient acquisition rules for model-based approximate Bayesian computation. arXiv:1704.00520 (3 April 2017).
31. H. Wang, J. Li, Adaptive Gaussian process approximation for Bayesian inference with expensive likelihood functions. arXiv:1703.09930 (29 March 2017).
32. J. M. Lueckmann *et al.*, "Flexible statistical inference for mechanistic models of neural dynamics" in *Advances in Neural Information Processing Systems 2017-December* (2017), pp. 1290–1300.
33. G. Papamakarios, D. C. Sterratt, I. Murray, "Sequential neural likelihood: Fast likelihood-free inference with autoregressive flows" in *International Conference on Artificial Intelligence and Statistics* (2019).
34. P. Ranjan, D. Bingham, G. Michailidis, Sequential experiment design for contour estimation from complex computer codes. *Technometrics* **50**, 527–541 (2008).
35. J. Bect, D. Ginsbourger, L. Li, V. Picheny, E. Vazquez, Sequential design of computer experiments for the estimation of a probability of failure. *Stat. Comput.* **22**, 773–793 (2012).
36. L. Heinrich, G. Louppe, K. Cranmer, excursion (doi:10.5281/zenodo.1634428) (2018).
37. M. Cutler, T. J. Walsh, J. P. How, "Reinforcement learning with multi-fidelity simulators" in *2014 IEEE International Conference on Robotics and Automation (ICRA)* (2014), pp. 3888–3895.
38. J. B. Hamrick *et al.*, Metacontrol for adaptive imagination-based optimization. arXiv:1705.02670 (7 May 2017).
39. K. Kandasamy, G. Dasarathy, J. Schneider, B. Poczos, "Multi-fidelity Bayesian optimisation with continuous approximations" in *Proceedings of the 34th International Conference on Machine Learning* (JMLR. org, 2017), vol. 70, pp. 1799–1808.
40. A. D. Gordon, T. A. Henzinger, A. V. Nori, S. K. Rajamani, "Probabilistic programming" in *Proceedings of the Conference on the Future of Software Engineering* (ACM, New York, NY, 2014).
41. A. Doucet, A. M. Johansen, "A tutorial on particle filtering and smoothing: Fifteen years later" in *Handbook of Nonlinear Filtering* (2009), vol. 12, p. 3.
42. A. G. Baydin *et al.*, Etalumis: Bringing probabilistic programming to scientific simulators at scale. arXiv:1907.03382 (8 July 2019).

43. M. M. Graham, A. J. Storkey, Asymptotically exact inference in differentiable generative models. *Electron. J. Stat.* **11**, 5105–5164 (2017).
44. J. Brehmer, G. Louppe, J. Pavez, K. Cranmer, Mining gold from implicit models to improve likelihood-free inference. *Proc. Natl. Acad. Sci. U.S.A.* **117**, 5242–5249 (2018).
45. M. Stoye, J. Brehmer, G. Louppe, J. Pavez, K. Cranmer, Likelihood-free inference with an improved cross-entropy estimator. arXiv:1808.00973 (2 August 2018).
46. A. G. Baydin, B. A. Pearlmutter, A. A. Radul, J. M. Siskind, Automatic differentiation in machine learning: A survey. *J. Mach. Learn. Res.* **18**, 1–43 (2018).
47. C. Olah, Neural networks, types, and functional programming (2015). https://www.facebook.com/yann.lecun/posts/10155003011462143. Accessed 1 January 2018.
48. Y. LeCun, Deep learning est mort. vive differentiable programming! (2018). https://colah.github.io/posts/2015-09-NN-Types-FP/. Accessed 1 January 2018.
49. T. A. Le, A. G. Baydin, F. Wood, "Inference compilation and universal probabilistic programming" in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017* (PMLR, Fort Lauderdale, FL, 2017), vol. 54, pp. 1338–1348.
50. D. J. Rezende, S. Mohamed, "Variational inference with normalizing flows" in *32nd International Conference on Machine Learning, ICML 2015* (2015), vol. 2, pp. 1530–1538.
51. G. Papamakarios, I. Murray, "Fast e-free inference of simulation models with Bayesian conditional density estimation" in *Advances in Neural Information Processing Systems* (2016), pp. 1036–1044.
52. R. Izbicki, A. B. Lee, T. Pospisil, ABC–CDE: Toward approximate Bayesian computation with complex high-dimensional data and limited simulations. *J. Comput. Graph Stat.* **28**, 481–492 (2019).
53. B. Paige, F. Wood, "Inference networks for sequential Monte Carlo in graphical models" in *33rd International Conference on Machine Learning, ICML 2016* (2016), vol. 6, pp. 4434–4444.
54. D. Tran, R. Ranganath, D. M. Blei, "Hierarchical implicit models and likelihood-free variational inference" in *Advances in Neural Information Processing Systems*, I. Guyon *et al.*, Eds. (2017), vol. 2017, pp. 5524–5534.
55. C. Durkan, G. Papamakarios, I. Murray, Sequential neural methods for likelihood-free inference. arXiv:1811.08723 (21 November 2018).
56. J. M. Lueckmann, G. Bassetto, T. Karaletsos, J. H. Macke, "Likelihood-free inference with emulator networks" in *Proceedings of The 1st Symposium on Advances in Approximate Bayesian Inference*, F. Ruiz, C. Zhang, D. Liang, T. Bui, Eds. (PMLR, 2019), vol. 96, pp. 32–53.
57. J. Alsing, T. Charnock, S. Feeney, B. Wandelt, Fast likelihood-free cosmology with neural density estimators and active learning. *Mon. Not. R. Astron. Soc.* **488**, 4440–4458 (2019).
58. R. M. Neal, "Computing likelihood functions for high-energy physics experiments when distributions are defined by simulators with nuisance parameters" in *Statistical Issues for LHC Physics. Proceedings, PHYSTAT-LHC 2007* (2007), pp. 111–118.
59. Y. Fan, D. J. Nott, S. A. Sisson, Approximate Bayesian computation via regression density estimation. *Stat* **2**, 34–48 (2013).
60. K. Cranmer, J. Pavez, G. Louppe, Approximating likelihood ratios with calibrated discriminative classifiers. arXiv:1506.02169 (6 June 2015).
61. O. Thomas, R. Dutta, J. Corander, S. Kaski, M. U. Gutmann, Likelihood-free inference by ratio estimation. arXiv:1611.10242 (30 November 2016).
62. M. U. Gutmann, R. Dutta, S. Kaski, J. Corander, Likelihood-free inference via classification. *Stat. Comput.* **28**, 411–425 (2018).
63. T. Dinev, M. U. Gutmann, Dynamic likelihood-free inference via ratio estimation (DIRE). arXiv:1810.09899 (23 October 2018).
64. J. Hermans, V. Begy, G. Louppe, Likelihood-free MCMC with approximate likelihood ratios. arXiv:1903.04057 (10 March 2019).
65. A. Andreassen, B. Nachman, Neural networks for full phase-space reweighting and parameter tuning. arXiv:1907.08209v1 (18 July 2019).
66. J. Brehmer, K. Cranmer, G. Louppe, J. Pavez, Constraining effective field theories with machine learning. *Phys. Rev. Lett.* **121**, 111801 (2018).
67. G. Louppe, K. Cho, C. Becot, K. Cranmer, QCD-aware recursive neural networks for jet physics. *JHEP* **2019**, 57 (2019).
68. A. Andreassen, I. Feige, C. Frye, M. D. Schwartz, JUNIPR: A framework for unsupervised machine learning in particle physics. *Eur. Phys. J. C* **79**, 102 (2019).
69. G. Carleo *et al.*, Machine learning and the physical sciences. *Rev. Mod. Phys.* **91**, 045002 (2019).
70. A. Munk *et al.*, Deep probabilistic surrogate networks for universal simulator approximation. arXiv:1910.11950 (25 October 2019).
71. J. Alsing, B. Wandelt, Generalized massive optimal data compression. *Mon. Not. R. Astron. Soc. Lett.* **476**, L60–L64 (2018).
72. J. Alsing, B. Wandelt, S. Feeney, Massive optimal data compression and density estimation for scalable, likelihood-free inference in cosmology. *Mon. Not. R. Astron. Soc.* **477**, 2874–2885 (2018).
73. P. De Castro, T. Dorigo, INFERNO: Inference-aware neural optimisation. *Comput. Phys. Commun.* **244**, 170–179 (2019).
74. J. Alsing, B. Wandelt, Nuisance hardened data compression for fast likelihood-free inference. *Mon. Not. R. Astron. Soc.* **488**, 5093–5103 (2019).
75. S. S. Wilks, The large-sample distribution of the likelihood ratio for testing composite hypotheses. *Ann. Math. Stat.* **9**, 60–62 (1938).
76. A. Wald, Tests of statistical hypotheses concerning several parameters when the number of observations is large. *Trans. Am. Math. Soc.* **54**, 426 (1943).
77. G. Cowan, K. Cranmer, E. Gross, O. Vitells, Asymptotic formulae for likelihood-based tests of new physics. *Eur. Phys. J. C* **71**, 1554 (2011).
78. F. Wood, J. W. Van De Meent, V. Mansinghka, A new approach to probabilistic programming inference *J. Mach. Learn. Res.* **33**, 1024–1032 (2014).