MAIN ARTICLE OPEN ACCESS

Incorporating Deep Learning Into System Dynamics: Amortized Bayesian Inference for Scalable Likelihood-Free Parameter Estimation

Hazhir Rahmandad[1] | Ali Akhavan[2] | Mohammad S. Jalali[2]

[1]Sloan School of Management, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA | [2]MGH Institute for Technology Assessment, Harvard Medical School, Boston, Massachusetts, USA

Correspondence: Hazhir Rahmandad (hazhir@mit.edu)

Received: 15 October 2024 | **Revised:** 6 December 2024 | **Accepted:** 27 December 2024

Funding: This work was supported in part by the U.S. National Science Foundation, Division of Mathematical Sciences and Division of Social and Economic Sciences (Award No. 2229819).

Keywords: amortized Bayesian inference | machine learning | neural networks | neural posterior estimation | parameter estimation | uncertainty quantification

ABSTRACT

Estimating parameters and their credible intervals for complex system dynamics models is challenging but critical to continuous model improvement and reliable communication with an increasing fraction of audiences. The purpose of this study is to integrate Amortized Bayesian Inference (ABI) methods with system dynamics. Utilizing Neural Posterior Estimation (NPE), we train neural networks using synthetic data (pairs of ground truth parameters and outcome time series) to estimate parameters of system dynamics models. We apply this method to two example models: a simple Random Walk model and a moderately complex SEIRb model. We show that the trained neural networks can output the posterior for parameters instantly given new unseen time series data. Our analysis highlights the potential of ABI to facilitate a principled, scalable, and likelihood-free inference workflow that enhance the integration of models of complex systems with data. Accompanying code streamlines application to diverse system dynamics models.

1 | Introduction

System dynamics modeling draws on various data sources, including qualitative, archival, and numerical (Forrester 1987) to build models of important problems. Without empirical support, theoretical claims or policy recommendations can be misguided (Sterman 2018). A continuous conversation between models and data helps develop better theory and policy (Popper 1934; Box, Hunter, and Hunter 1978; Homer 1996) and ensures models' relevance.

For example, Rahmandad and colleagues developed system dynamics models to understand heterogeneity in COVID-19 deaths, predict future trajectories, and assess vaccine benefits (Rahmandad, Lim, and Sterman 2021; Rahmandad and Sterman 2022; Rahmandad, Xu, and Ghaffarzadegan 2022a). The model(s) went through over 80 versions. Each iteration compared model outputs with empirical data (deaths, cases, hospitalizations, excess mortality) and estimated parameters against the literature (e.g., vaccine effectiveness, immunity loss time, and infection fatality rates). Discrepancies motivated searches

Accepted by Andreas Größler

for underlying causes and mechanisms, which then informed model updates from behavioral response feedback and exploring adherence fatigue to loss of immunity, vaccination dynamics, separation of populations based on prior immunity status, and many others. Absent this iterative process, earlier models would have produced different, likely inferior, and even misleading conclusions and recommendations. Therefore, iterative refinement of models is vital for ensuring the reliability and relevance of model-driven insights in addressing real-world challenges.

Enhancing models iteratively involves three key elements: feedback to refine model structures, parameter identification, and a signal to know when the model is good enough for the purpose at hand. Historically, iterative modeling relied on comparing models to qualitative and archival data (Forrester 1987). While revealing structural shortcomings, these methods do not assess parameter accuracy or overall model quality. Even with numerical data, traditional methods focused on hand calibration of point estimates and simple statistical tests (Lyneis and Pugh 1996; Sterman 2000). Automated calibration methods have since emerged, simplifying the search for optimal parameters (Oliva 2003).

Recent work aims not only to find point estimates but also to quantify uncertainty in model parameters (Jalali, Rahmandad, and Ghoddusi 2015; Andrade and Duggan 2021). Uncertainty quantification is critical because it allows researchers to determine if qualitative conclusions are statistically significant (Gelman et al. 1995; Kennedy 2008). It is also essential for projecting future trajectories and designing policies (Manski 2013), for example, the decision to purchase insurance is typically based on low probabilities of adverse events in the tail of outcomes distributions, not the most likely scenarios. Thus, quantifying uncertainty not only enhances model reliability and theoretical conclusions but also supports better decision-making.

The explosion in the availability of numerical data (Varian 2014; Blei and Smyth 2017) has made formal parameter estimation and uncertainty quantification widely feasible and expected across many disciplines. Classical methods for uncertainty quantification often rely on explicitly defining the 'likelihood' of some set of model parameters given an observed dataset. When feasible, these methods provide important advantages in simplicity, computational costs, transparency, and efficiency (Gill 2002; Casella and Berger 1990). The parameter combination that maximizes the likelihood of generating the observed dataset becomes the maximum likelihood estimate, and the curvature of the likelihood function around the maximum likelihood point informs uncertainty in estimates. Whether that curvature is approximated using the Hessian matrix (of likelihood with respect to model parameters) or empirically sampled using Markov Chain Monte Carlo (MCMC) and related methods, one can quantify uncertainty in parameters when likelihoods are available (Gelman et al. 1995).

Unfortunately, likelihoods are not available for most system dynamics models due to nonlinearity, process noise, and high-dimensional parameter spaces. Absent likelihoods, some may opt for simplifying the model to enable explicit likelihood calculations (Box and Jenkins 1976), trading off model quality for tractability. Another approach is to use approximations of the likelihood function that may be inaccurate but flexible enough to quantify parameter uncertainty (Li, Rahmandad, and Sterman 2022). Such approximations, when effective, are appealing but should be designed and validated for each case. A third approach is to use more complex state resetting and filtering methods (e.g., Kalman or particle filter) to enable better estimates of true likelihood, albeit at increased computational costs (Arulampalam et al. 2002; Eberlein 2015).

However, methods exists that do not require likelihoods, including simulation-based inference approaches, spanning method of (simulated) moments, variational inference, approximate Bayesian calculation, and related approaches (Hansen 1982; Marin et al. 2012; Jalali, Rahmandad, and Ghoddusi 2015; Hosseinichimeh et al. 2016; Blei, Kucukelbir, and McAuliffe 2017). These methods often (but not always, see Drovandi and Frazier 2022) calculate some informative summary statistics of the data and search over model parameters for values that offer summary statistics matching those in the data. These methods do not usually have the efficiency of likelihood-based methods (i.e., they lose information in the estimation process and thus estimate wider credible intervals). Part of the inefficiency is due to the ad-hoc nature of selecting the summary statistics. These methods are also usually computationally less efficient. These limitations add up, limiting many likelihood-free methods to simpler models (e.g., a dozen unknown parameters on models that simulate in a fraction of a second) and good summary statistics can be specified manually (but see variational inference Blei, Kucukelbir, and McAuliffe 2017).[1]

In short, parameter estimation and uncertainty quantification for system dynamics models are increasingly important and can benefit from advances in other fields. A robust solution enables the estimation of both the model parameters and their uncertainty and validation of our estimation framework. Calibration methods most common in system dynamics literature offer point estimates for model parameters but fall short on other criteria for a principled inference workflow. This paper offers a bridge to rapid advances in estimation methods in neighboring fields (Vehtari, Gelman, and Gabry 2017), including those building on machine learning methods that leverage neural networks to provide more comprehensive solutions to estimation problems.

In the next section, we provide a high-level overview of this evolving methodological toolbox. We then apply one promising method from this set to two simple system dynamics models to assess its viability and promise. We demonstrate how a SD model can generate the synthetic data needed to train neural networks that enable Bayesian inference, and how new synthetic datasets (i.e., data with known true parameter values) can be used for validation of the inference process. Two case studies demonstrate the practical implications and benefits of integrating machine learning techniques into system dynamics.

2 | Neural Networks for Estimating Model Parameters

Recent advances in machine learning methods leveraging neural networks (NNs) are quickly changing the landscape for parameter estimation methods (Raissi, Perdikaris, and

Karniadakis 2019; Cranmer, Brehmer, and Louppe 2020). Recent work shows NNs can learn the posterior distribution (i.e., the probability distribution of model parameters conditioned on observed data) of parameters without requiring an explicit likelihood function (Tran et al. 2019; Papamakarios et al. 2021; Kingma and Welling 2022). These methods can overcome challenges due to intractable likelihood functions and may even reduce the computational costs associated with iterative simulation and optimization processes traditionally required for parameter estimation.

The details of implementing NNs vary depending on the specific method used and are reviewed elsewhere (e.g., see Cranmer, Brehmer, and Louppe 2020; Papamakarios et al. 2021). The core problem is seen as identifying the (posterior) distribution of the (vectors of) model parameters (θ) given an observed dataset (x). Two of the most common categories of these methods include Neural Ratio Estimation (NRE) (Durkan et al. 2019; Hermans, Begy, and Louppe 2020) and Neural Posterior Estimation (NPE) (Lueckmann et al. 2017; Greenberg, Nonnenmacher, and Macke 2019) methods, while other approaches (such as synthetic likelihood estimation) are also available and rapidly evolving.

In NRE methods, a neural network receives combinations of θ and x as input and is trained to tell apart if the x has come from the θ that generated it (correct matching of parameter and data) or not (e.g., by scrambling which θ generated x, and showing incorrect matches to NN). Thus, posterior estimation turns into a classification problem in which NNs excel. Once the NN has learned to make this classification, it has implicitly learned the posterior: Given a data set x, what is the likelihood of different parameter combinations having generated that dataset? With this information embedded in the NN, one can rapidly sample from the posterior using methods like MCMC without requiring sampling from the (computationally expensive) simulation model.

The NPE methods attempt to learn the full posterior distribution directly. A common approach is to train a reversible NN (one where the NN's transformation function can be reversed analytically, e.g., Normalizing Flows (Tabak and Turner 2013)) so that inputting both x and θ (that generated x) into the NN, the network (on its "forward" path) learns to output a simple distribution (e.g., a standard multivariate Gaussian) of the same dimensionality as θ. This "training phase" process leverages the inherent flexibility of neural networks to approximate complex, non-linear mappings between inputs and outputs. In the inference phase, by inverting that network analytically and conditioning it on an observed dataset x, one can give it samples of standard multivariate Gaussian distribution and produce, at the end of the "inverse" path, samples from the posterior of the parameter distribution consistent with the dataset x. The task of training the network is pursued using an error function minimizing the gap between the outputs of the forward network and a standard multivariate Gaussian distribution, typically using variants of Kullback-Leibler divergence between the two distributions. Figure 1 illustrates such a workflow. An important

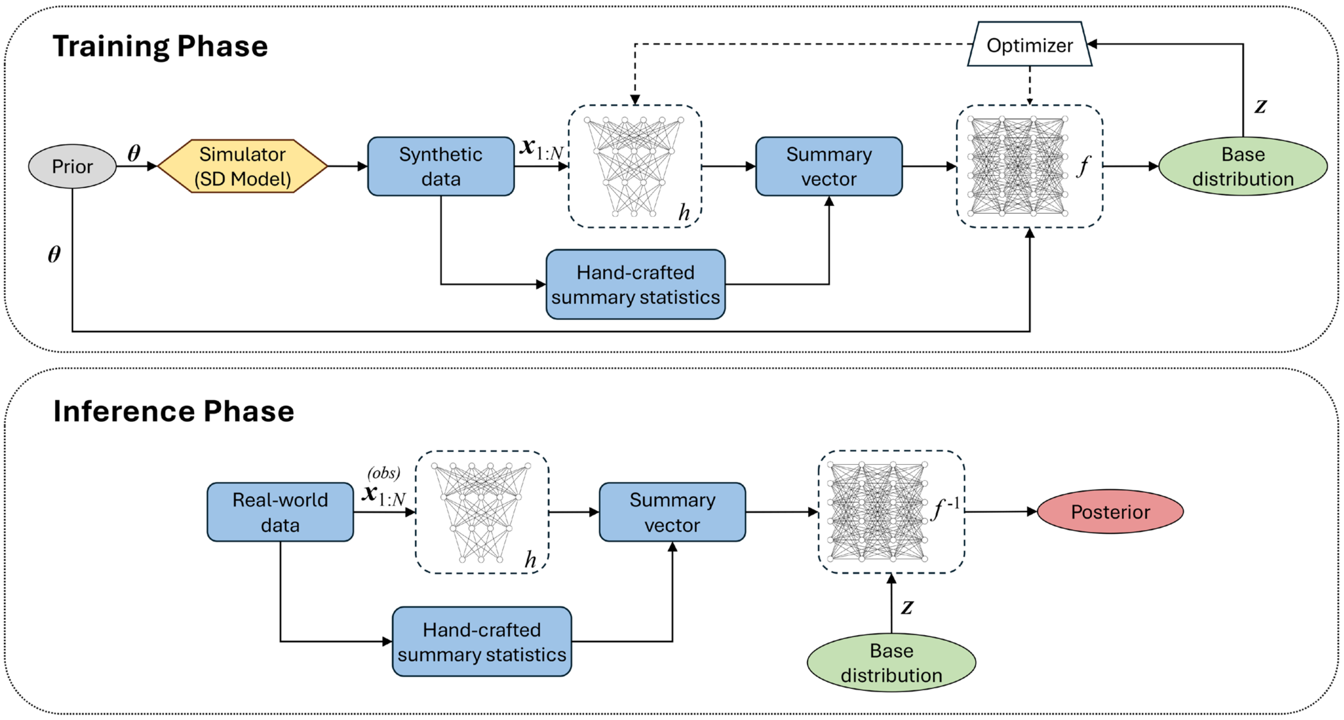


FIGURE 1 | Summary of the training and inference phases. During training samples of θ from prior distribution are fed into the system dynamics model to generate synthetic (simulated) data. Each synthetic dataset is turned into a summary vector using a summary neural network h and potentially augmented by hand-crafted summary statistics. The summary vector and original parameters are then fed into the inverse inference network f and an optimizer estimates the weights of h and f so that the output of inverse inference network converges to a simple base distribution (e.g., multi-variate Gaussian with dimensionality of θ). In the inference phase, an unseen (e.g., real-world) dataset is used to generate the summary vector leveraging estimated h. The inference network then uses the summary vector and samples from the base distribution to generate the posterior distribution. The figure is borrowed from Radev et al. (2023) and revised to match the specifics of this article.

feature of this method is the ability to generate samples from the posterior distribution at almost no cost: one simply feeds into the inverse network (analytically derived from the estimated forward NN) a sample from the multivariate Gaussian distribution and receives on the other end a sample of the posterior, bypassing the potentially slow and expensive MCMC step.

Both NRE and NPE methods offer two important features. First, they could include an NN that transforms simulation data into a set of summary statistics before feeding that to the main inference network. This preprocessing step is crucial for condensing the data into a manageable and meaningful form that captures the essence of the dataset. The user should decide on the dimensionality of summary statistics (which should be more than the number of estimated parameters, often by a factor of 2-4, to provide sufficient information about the data features). The summary network will then be trained alongside the inference one to generate informative summary statistics without the need for relying solely on hand-made statistics, making the method fully automated. Second, by inputting different θ and x values, the network learns not only the posterior for a given x but also the posterior for any dataset that could come from the range of θ for parameters (priors in a Bayesian setup) that it has been trained on. This feature leads to "amortized" Bayesian inference (ABI) of models: we could fully solve the inference problem for a whole family of models (same structure, different θ values coming from a prior distribution) rather than a single dataset. Amortization refers to the efficient reuse of computational resources, spreading out the initial computational cost of solving the inference problem over multiple model instances. It overcomes the scalability challenges often faced in Bayesian inference, offering a practical solution for complex modeling scenarios when the same model will be reused (for different instances, subjects, etc.). Once such amortized estimation is complete, the parameter posteriors for *any* dataset can be obtained instantly and at little cost. This could be a huge benefit when the estimation work is not one-off. In such cases, the amortization would save in the order of the number of required re-calibrations.

Training an NN that learns the posteriors for *all* different datasets (generated from a prior distribution) could be computationally expensive, and therefore, "sequential" methods for NRE and NPE have been developed where the sampling distribution from possible θ is narrowed down adaptively: as we learn the likely (posterior) distribution of parameters given the target dataset x, we focus on learning the posterior using samples from this more limited parameter space. This adaptive approach optimizes the learning process by concentrating computational efforts on the most relevant parts of the parameter space. The final network would then be primarily applicable to the target dataset that could be estimated more efficiently (Lueckmann et al. 2021). While ABI may seem dauntingly complex, the task has proved easier in practice than the combinatorial explosion of the input data space may suggest. For example, the initial training time may increase by one to two orders of magnitude compared to the relevant sequential method, but future inference will be almost instantaneous (Radev et al. 2022). This efficiency comes from the fact that the NN learns well, and becomes more robust, from being trained on a larger set of input parameters and outputs, and thus can better identify viable vs. unlikely parameter combinations.

Amortizing inference offers two notable benefits. First, once trained, the parameter posteriors for *any* dataset can be obtained instantly and at little cost. The savings would be great when the estimation work is not one-off; For example, if one needs to estimate a model for different experimental subjects who have different data series using the same underlying model. Second, amortization enables methods for assessing the reliability of inference (Gershman and Goodman 2014; Gelman et al. 2020). These methods require going through the inference process multiple times for different datasets. For example, consider the credible intervals (CIs) generated for one of the model parameters. How do we know if those CIs are reliable? One approach is to generate many synthetic datasets from the model using (known) parameters that are close to those estimated for our empirical case, then estimate the CIs for these new datasets, and finally assess if the fraction of ground truth parameters that fall within corresponding CIs is consistent with theoretical values. For example, we expect about 90% of ground truth parameters to fall within their corresponding 90% CI. To test this expectation, we need many separate estimations, which are fast with ABI and costly without.

Other numerical validation methods for inference similarly benefit from ABI. For example, Simulation-Based Calibration (SBC) (Talts et al. 2020) relies on a simple theorem: If we sample from a prior distribution for θ, generate a dataset x for each θ, and then sample from the posterior of θ given the realized x, we will get back to the prior distribution of θ. Comparing the initial prior and the resulting distribution (which should be the same as the prior) will inform whether the estimated posterior is correct, or the inference is problematic. SBC starts with many samples from θ (computationally trivial), generates an instance of x for each θ (a simulation; computationally cheap), then samples from the posterior of θ given x (computationally cheap with ABI, but very expensive with non-amortized methods). Thus, absent ABI, SBC and many other empirical methods for validating inference are too expensive for all but the simplest models.

Overall, ABI methods have promising features. They enable inference with only a simulation model and absent likelihoods, a situation common to system dynamics modeling. Moreover, whereas non- amortized methods require a separate estimation for each new dataset, with ABIs a single run through the training phase enables instant estimation and generation of samples from parameter posteriors for a large number of datasets. Those dataset may come from different subjects in experiments, different firms in a dataset, and so on. This opens up the path to using dynamic models for populations of units (people, firms, countries, etc.) more efficiently. They also provide methods, such as SBC, for validating the inference framework. Are these methods able to tackle typical estimation challenges in system dynamics research and practice? This is the question we address in the current paper.

3 | Study Design

In this study, we focus on assessing the applicability of neural estimation methods to system dynamics models. Reviewing prior literature, we focus on an ABI method from the NPE category, which offers a state-of-the-art performance in this space

(Cranmer, Brehmer, and Louppe 2020) with a user-friendly Python package, BayesFlow (Radev et al. 2022). We study viability (could this method work at all for system dynamics models?), scalability (how large a model/parameter space is feasible to tackle?), and ease of use (could the methods be packaged so that a typical user can benefit from them with basic programming skills?).

Given limited space, we focus on two models, Random Walk and SEIRb. The first is very simple, with a first-order autocorrelated noise structure with a drift and measurement error. This model enables the systematic identification of issues related to the impact of process and measurement noise. The second model is the classical SEIR plus a feedback loop for the impact of recent deaths on risk perception and, thus, contact, following Rahmandad, Xu, and Ghaffarzadegan (2022b). While still simple, with 12 parameters and reference modes spanning exponential growth, overshoot and collapse, and overshoot and oscillation, SEIRb offers a more challenging estimation task.

Traditional estimation methods focus on minimizing the gap between simulated model outputs and empirical data by changing model parameters. Deep learning's application to estimation takes a somewhat different route: it focuses on learning the patterns in data to map how parameter inputs relate to model outputs for a wider range of input-output combinations. In system dynamics, typical inputs are the model parameters, and outputs are often simulated time series for a subset of model variables. Once a neural network has learned the input-output mapping, it can take a set of outputs and infer the parameters (inverse problem) responsible for generating those outputs. Therefore, a neural network can be trained by samples of parameter inputs and simulated model outputs, before it needs to utilize the empirical dataset(s) at hand. In short, synthetic data (samples of parameters and simulation outputs) are all that is needed for training the inference neural networks, with the actual inference step for an empirical dataset becoming trivially easy: you provide the inference network with your dataset, and it outputs the posterior for model parameters.

Synthetic data plays another important role in validation of inference methods (Nikolenko 2021; De Melo et al. 2022). Specifically, for empirical data, we do not know the "true" parameter values that generated the data. Therefore, we have no way of assessing the quality of inference: how do we know if inferred posterior includes true parameters if we do not know those ground truths? The standard solution to this challenge is to conduct inference on synthetic "validation" data because that data comes from simulations with known "ground truth" parameters. In this step, the neural network is blinded to the ground truth parameter values in the validation dataset; thus, the inferred parameter values (and posteriors) may be far from the ground truth values, alarming the analyst to deficiencies in the inference process. Combined with ABI, this validation method is especially powerful because one can quickly generate a large synthetic validation dataset and conduct the inference rapidly on all those distinct problems to assess the overall quality of the inference method. Not only standard practice in statistics and machine learning (Lueckmann et al. 2021; Radev et al. 2022), system dynamics scholars have also used synthetic data to validate their estimation workflow (e.g., see Rahmandad, Lim, and Sterman 2021).

In the rest of the manuscript, we provide an overview of the estimation process using the BayesFlow package, elaborate on the two test models, apply the tools to these models under different hyperparameters that control inference, and report on the efficacy of methods, effective NN training settings, and computational costs.

4 | Methods

A robust solution to the problem of integrating data and models allows us to incorporate qualitative data into our estimation workflow, enables the estimation of model parameters and their uncertainty in light of data, helps us validate our estimation framework, and can signal when more iterations on the model are called for and when we have a satisfactory solution. A principled inference workflow ensures these requirements are explicit and can be addressed in different steps of the process. Such workflows have been developed with different points of focus; here, we adopt some of the terminology and steps from an increasingly standard Bayesian inference framework (Gelman et al. 2020) and adapt them to working with system dynamics models. We overview the key steps of the inference framework and then introduce the components we use in our analysis.

4.1 | Inference Steps

The inference process includes defining a generative engine, the neural networks for estimation, and the training schedule for inference.

4.1.1 | Defining the Generative Engine

Estimation starts with defining a "generative engine" which consists of three components: (1) Our system dynamics simulation model. (2) The variables in the simulation model for which we have real-world data (i.e., an observation set). And (3) A prior distribution on unknown model parameters. Priors act as a synthesis of our pre-existing knowledge and assumptions, guiding the estimation process within plausible bounds. Note that we often also have parameters that we know with good certainty (e.g., the total population of a simulated country) or assume (e.g., using a linear function assumes a power of "one") and thus are not part of the prior. Prior plays two roles in the workflow. First, it limits estimation outcomes to what is feasible based on the physics of the problem, qualitative data, and prior theory. Second, by sampling from the prior and simulating the model, one creates a wide range of behaviors that defines the whole category of system behaviors feasible under the generative engine rather than a single reference mode (Forrester 1961). The ABI aims to estimate parameters coming from any combination in this space. To keep the exposition simple, we use less informative (uniformly distributed) priors. This may also create more diverse outcomes when sampling from the prior, which are harder to learn by NNs; it is easy to replace these with other prior distributions.

The simulation model should not only include the deterministic processes we want to capture, but also the stochastic ones that drive a wedge between the perfect model and observed outcomes in the real word. That will allow inference to learn how to interpret the magnitude of the noise and associate it with relevant noise parameters. The distinction between process and measurement noise may be helpful for thinking about such stochasticity, though they are similarly treated in the workflow. Process noise incorporates randomness that changes the dynamics of the system but is not deterministically known (e.g., weather effects in a model of COVID-19), while measurement noise captures uncertainty in the measurement process that otherwise does not change the dynamics of the system (e.g., the measurement errors in surveys). Both types should be explicitly modeled as part of the simulation model, with their corresponding free parameters (e.g., standard deviation, auto-correlation) included in our estimated parameters and the prior distributions. As a result, typical models going into generative engines are stochastic, a departure from standard practice in classical system dynamics.

4.1.2 | Defining Summary and Inference Neural Networks

Two different neural networks will be used in the ABI process we discuss here. The summary network converts the output of the simulation model into summary statistics to be used by the inference network. The inference network takes those summary statistics and outputs the posteriors for the parameters. Different types of neural networks could be used in both, though research is ongoing to identify better alternatives for different types of problems. We focus on using default options implemented in the BayesFlow that are most suitable for the types of problems system dynamicists may encounter and briefly note the relevant hyperparameters, which could be adjusted for each problem based on the cumulative knowledge of what works in this space.

One effective summary network architecture for time series data combines a sequence of multi-layer 1D convolutional networks followed by a Long Short-Term Memory (LSTM) network (Radev et al. 2021). While much detail could be customized, the most important hyperparameters for such "SequenceNetwork" architectures are the number of convolutional layers, the number of LSTM units, and whether the network digests the data in a "bidirectional" fashion. For any summary network, we should also specify the dimensionality of the summary statistics (network output), which should be larger than the number of parameters (~2–4 times may be a good heuristic from our experience). Alternatively, or additionally, "manual" summary statistics could also be incorporated smoothly. We will discuss one example in our analysis.

For the inference NN, multiple layers of an invertible network with common architectures, including "affine" (Kingma and Dhariwal 2018; Ardizzone et al. 2021) and "spline" (Durkan et al. 2019) (or their combinations, i.e., "interleaved") are common. Whereas affine networks are computationally more efficient to train, spline networks may be more expressive for complex geometries of posteriors. The architecture and the number of coupling layers are the primary hyperparameters, though we only focus our comparisons on the number of coupling layers, going with the default "affine" architecture that is computationally more efficient and thus more scalable for higher dimensional estimation problems common to system dynamics practice. Table 1 presents key hyperparameters for summary and inference networks.

4.1.3 | Specifying the Training Schedule

In BayesFlow, the training of neural networks is automated under the hood, leveraging the TensorFlow framework and packages. All an analyst needs to do is to write the code to receive proposed model parameters from BayesFlow (drawn from the prior), do the simulations in the software of the analyst's choice (in our case, Vensim), and return the results to BayesFlow, in a fast and automated fashion. Training happens by minimizing a loss function that measures the Kullback-Leibler (KL) divergence between the output of the forward pass on the invertible network (after receiving inputs from the summary network) and the standardized multivariate Gaussian distribution. The training schedule specifies how many simulations are to be conducted and how they should be leveraged in training the NNs. Training happens through stochastic gradient descent methods, where the gradients of the loss function with respect to NNs parameters are calculated on a "batch" of simulations, and NN parameters (i.e., weights and biases) are then updated with a "learning rate." For a given dataset of simulations, training may proceed by dividing all simulations into the required iterations for a single epoch of going through all the data (see explanations in Table 1). For example, 1,024 (=64×16) simulations could be divided into 64 iterations per epoch with a batch size of 16 simulations per iteration. Larger batches give more precise gradients but take more time to calculate a single gradient. The learning rate is often dynamically adjusted during the training and is reduced through a "decay" strategy (e.g., cosine decay) so that by the end of the training, the NN parameters are fine-tuned, and the network is stabilized. Multiple epochs could be defined for training, and as long as loss values on a validation dataset (to avoid overfitting) decline over epochs, training on the existing data adds value. While such "offline" training on a fixed dataset simulated at the outset is conceptually straightforward, other alternatives may work better in some settings. For example, if the simulation model is fast and data exchange between simulation and NN training is quick, fully online training may be preferred, where every batch calls for new samples (and overfitting is not a concern). A round-based mixed approach goes through multiple rounds; at the beginning of each, a new dataset of simulations is created and appended to the previous ones, and then the training goes through the required iterations, given the batch size, for the number of epochs specified per round. This method incorporates new data continuously but does not discard the old data either.

4.1.4 | Implementation Notes

All simulations are run in Vensim DSS, and data is transferred to BayesFlow using either DLL functionality (more efficient for larger models; simplified with VenPy package) or automated scripts (using VST package). To maximize efficiency, we use sensitivity analysis in Vensim that could be parallelized and

TABLE 1 | Key hyperparameters and their definitions.

Category	Hyperparameter	Explanation
Summary network	Convolutional layers	A computational layer used in neural networks that processes data through a series of learnable filters. This layer helps capture spatial or temporal hierarchies in data by applying convolutions over the input and passing the result through an activation function.
	Summary dimensions	The size of the output vector produced by the summary network. It represents the condensed information extracted from the input data, which is then fed into the inference network.
	LSTM units	(Number of) basic units of a Long Short-Term Memory (LSTM) layer, designed to remember information for long periods. LSTM units help process time-series data by capturing temporal dependencies and sequences in the input.
Inference network	Coupling layers	(Number of) layers of normalizing flow network which learn the shape of posterior distribution under different datasets. They allow the model to perform intricate transformations by coupling parts of the input reversibly, aiding in efficient inference.
	Bidirectional	Whether to use bidirectional LSTMs that allow the network to process data in both forward and backward directions. This improves the model's understanding of early data and utilizing it more effectively.
Training	Learning rate	The learning rate determines the size of the steps taken while optimizing the network's weights. A higher learning rate may lead to faster convergence but can become unstable while a lower learning rate ensures more stable but slower convergence.
	Batches	The number of training samples processed to calculate gradients before the NN's internal parameters are updated. Batch size affects the speed and stability of the learning process, with smaller batches generally providing more frequent but less reliable updates and larger batches reducing the number of updates but making each more precise.
	Epochs	One complete pass through the current training dataset. The number of epochs determines how many times the learning algorithm will work through the entire dataset. More epochs allow the model to better learn from the data at the cost of increased computational time and potential overfit.

compiled, with proposed BayesFlow parameters written in a text file that is input to sensitivity analysis. We have written the Python functions to integrate these capabilities with Vensim seamlessly, and the code is available for others to use and extend to new problems (see the GitHub link in the Supporting Information). With this machinery in place, the application costs are rather minimal after an initial installation process that may take a couple of hours, and most users do not need much coding to leverage the tools (and what coding is needed could be handled with the support of increasingly capable generative AI platforms such as ChatGPT and Gemini). It is straightforward to use other simulation software instead of Vensim, if they could receive programmatic instructions to generate a batch of simulations and output it back in text or DLL connections. In practice, the simulation time may not be too long for most system dynamics models, where a few million simulations should suffice for ABI; rather, the training time for NNs becomes the main computational bottleneck, and there the use of GPUs proves helpful, cutting training times by a few folds compared to CPU-based training. All reported analyses were conducted on a Windows environment using an NVIDIA A2 Tensor Core GPU with 16GB GPU Memory. Although the PC has abundant CPU (128 Cores) and RAM (512GB) resources, the BayesFlow algorithm barely used more than 2% of such resources compared to more than 95% utilization of GPU during our experiments.

4.2 | Inference Validation and Assessment

Once the NN is trained, the posteriors for any dataset can be (almost) instantaneously generated. We need metrics to quantify how effective this process is. The metrics we explore below focus on different costs of inference and the internal consistency of the inference process, that is, if the model was correct, how well could we identify the parameters and their uncertainty?

4.2.1 | The Data and Computational Requirements for Estimating the Model

How many simulations are required for satisfactory training? How much training time is needed for a satisfactory output? One could change the data/training size and observe the impact on performance or fix the data/training time and measure the change in performance. For data input, we focus on the number of simulations conducted, and for computational costs, we report wall time along with hardware specifications.

Other useful measures include posterior contraction and Z-score. The former measures the shrinkage fraction in the standard deviation of parameters going from the prior distribution to the estimated posteriors. A value close to one suggests that inference has extracted a lot of data and provides reliable estimates for the parameter. Posterior Z-scores normalize and compare the estimated parameter values against the ground truth, with a value of 0 offering perfect recovery and an expected spread with a standard deviation of one for a good inference.

4.2.2 | Quality of Inference

Multiple metrics could inform the quality of inference. The loss function from the training of neural networks is a useful indicator, especially when applied to a validation dataset not used in training (and thus not suffering from overfitting). While the KL divergence value is not fully comparable across problems, for a given problem, the loss values are directly comparable, and typically, values below 0 start to indicate good convergence.

SBC provides a more rigorous method for assessing inference quality (Talts et al. 2020). The inference is deemed reliable if the prior distribution for each parameter is similar to the samples of posteriors from different synthetic datasets generated (by the simulation model) after drawing the parameters from the prior distribution, generating data, and inferring posteriors. By generating hundreds of different parameter sets, many datasets per each parameter set (which would be different given the stochasticity of models), and conducting inference on all those data, one can use Empirical Cumulative Distribution Function plots (ECDF) (Säilynoja, Bürkner, and Vehtari 2022) to formally test the quality of inference.

The precision of estimated CIs is also intuitive and informative. We first create different synthetic datasets from ground truth parameters drawn from the prior distribution. Next, we estimate the posteriors and CIs for those and assess the empirical fraction of ground truth falling into different CIs, comparing that to the expected fraction. A 45° line suggests a good calibration of CIs.

4.3 | Model Assessment

Finally, model assessment could be pursued based on a host of metrics. Many focus on the predictive performance of the models out of sample, a useful measure, though one with caveats related to loss of precious data and limited information on how much room for improvement there may be. Another useful set of measures identifies the gap between empirical summary statistics and those coming from an estimated simulation model. Aggregating the gaps into a single measure, such as Maximum Mean Discrepancy (MMD) (Gretton et al. 2012), provides formal tests for assessing the quality of the model in light of the observed data and can inform future iterations (Schmitt et al. 2022). Due to limited space, we will not be utilizing this approach in the current paper and will only introduce the idea, given its relevance for an iterative inference workflow.

4.4 | Model 1: Random Walk

Figure 2 shows the structure of the Random Walk model. This model is, by design, very simple. It focuses on capturing process and multiplicative measurement noise which are sufficient to rule out explicit likelihood functions, but otherwise lacks many complexities common to SD models. The state (S) of the system evolves over time through processes known as drift (d) and shock (K). Drift (d) represents the predictable or "expected" change, while shock (K) represents the stochasticity, process noise, in the dynamics. Shock (K) is modeled as a normal distribution with a process noise standard deviation of σ_p. In addition, the state observed (S_{Obs}) is derived from the "true" state (S) with a multiplicative normally distributed measurement noise characterized by a standard deviation of σ_m. All model parameters (i.e., d, S_0, σ_p, and σ_m) are drawn from uniform distributions, with their corresponding minimum, maximum, and pseudo-random number stream (ns) (the additional links from *ns* to the parameters are hidden for visual clarity). Table 2 summarizes the model formulations and priors. Online Appendix S1 includes full model documentation.

Figure 3 shows the model's different modes of behavior as a result of unique sets of parameters generated through random draws from the uniform distributions of the priors.

4.5 | Model 2: SEIRb

Figure 4 shows the simplified structure of the SEIRb model. The model is similar to the classical SEIR (Susceptible, Exposed, Infectious, Recovered) model with an additional endogenous behavioral-risk response mechanism where transmission intensity grows (declines) as the death rate declines (grows). The

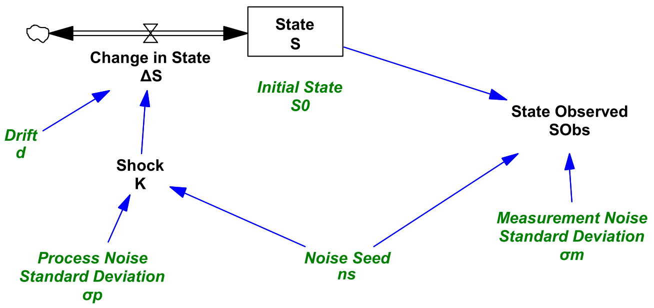


FIGURE 2 | Structure of the Random Walk model.

TABLE 2 | Parameter priors of the Random Walk model that are estimated.

Name	Prior	Units
Drift	$d = \text{Uniform}(-1, 1)$	1/Time
Initial state	$S_0 = \text{Uniform}(0, 10)$	Dimensionless
Process noise standard deviation	$\sigma_p = \text{Uniform}(0, 0.5)$	1/Time
Measurement noise standard deviation	$\sigma_m = \text{Uniform}(0, 0.5)$	Dimensionless

details of the model are described elsewhere (e.g., Rahmandad, Xu, and Ghaffarzadegan 2022b). In short, the Transmission Intensity (β) regulates the speed of transmission, and Patient Zero Arrival Time (t_0) specifies the introduction time for the disease. The Infection Fatality Rate (IFR) is the fraction of infected people who die. Time to onset and removal are assumed to be known based on clinical knowledge and not separately estimated. Besides these core SEIR components, a few parameters regulate the behavioral feedback loop. Two time constants, Time to Onset (t_s) and Time to Removal (t_r), regulate the adjustment of the perceived risk of death (PDR) to the actual death rates using an asymmetric first order smooth. Two parameters, Sensitivity to Death (α) and Death Risk Diminishing Impact (γ), regulate the strength of behavioral response (Effect of Perceived Risk on Attack Rate, EPA) using the following function:

$$\text{EPA} = \frac{1}{1 + (\alpha \cdot \text{PDR})^{\gamma}}$$

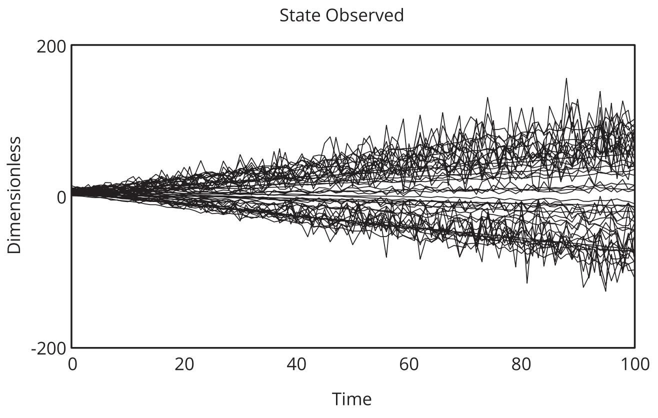


FIGURE 3 | Random Walk model behavior (50 simulations) using randomly drawn values from parameter priors.

We assume three data series are observable: Onset, Recovery, and Deaths. These measurements are imprecise, and part of the model's stochasticity comes from the multiplicative measurement noise affecting Simulated Onset Data (SOD), Simulated Recovery Data (SRD), and Simulated Death Rate Data (SDD)[2]. The stochasticity is also caused by the Process Noise (PN) impacting the Exposure Rate (ER). These noise functions all include Gaussian distributions with their corresponding (unknown) standard deviations (σ_i; for a total of 4 parameters), with PN being also first-order autocorrelated and thus including another unknown parameter (Process Noise Correlation Time, Crr_τ). Overall, the model includes 12 unknowns (to be estimated parameters), two assumed/known parameters, and three daily observed time series that inform inference.

Figure 5 shows the model's behavior generated using different parameter values randomly drawn from the uniformly distributed priors. The simulations represent different modes of behavior (e.g., overshoot and oscillation) for onset, recovery, and death rate data. For simplicity, we assumed an identical population (one million) across all simulations. All parameter priors, including the noise standard deviations and correlation time, are randomly drawn from their corresponding uniform distributions and used in the neural estimation. Table 3 summarizes the model parameters being estimated and their priors. We set the parameter boundaries based on the feasible empirical ranges in the case of COVID-19 for different parameters (e.g., Rahmandad, Xu, and Ghaffarzadegan 2022b). Online Appendix S1 includes full model documentation.

5 | Results

In reporting the results, we start by discussing experiments on various hyperparameters of the inference algorithm. Specifically,

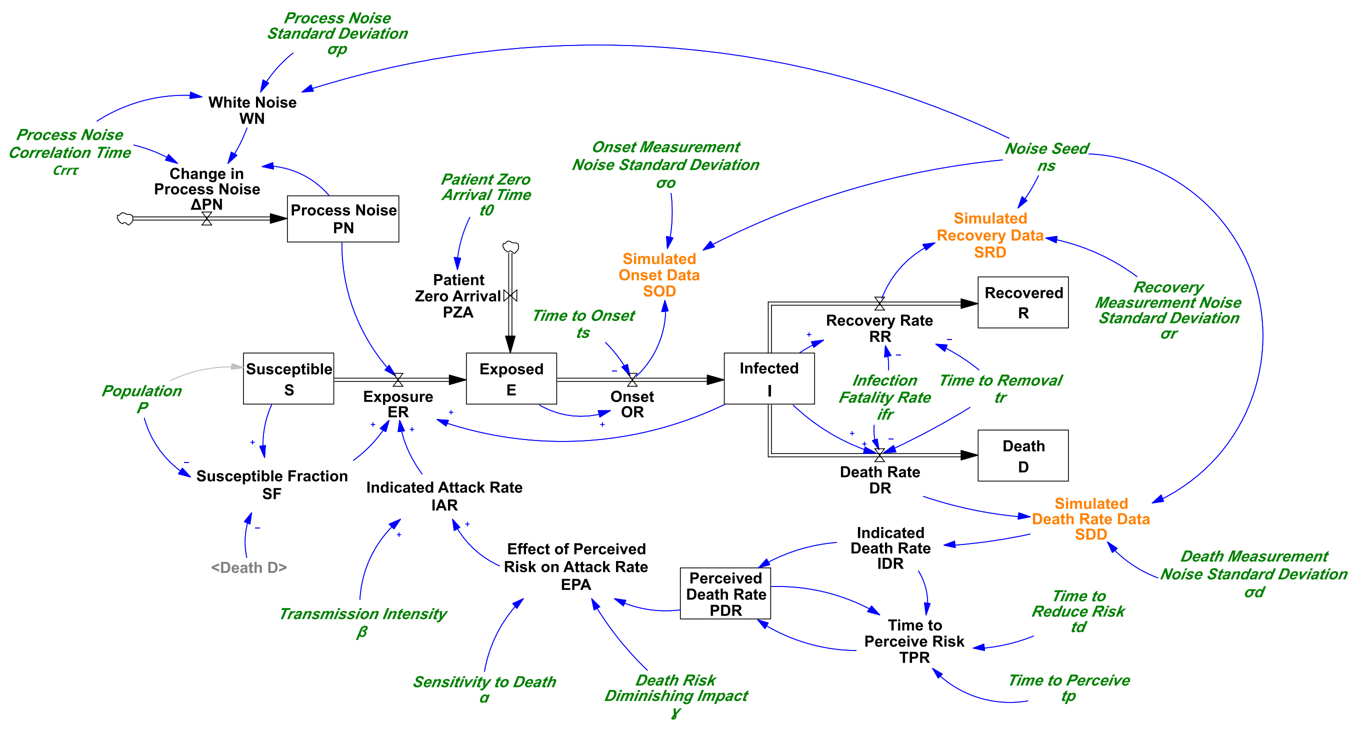

Process Noise Standard Deviation σp
White Noise WN
Noise Seed ns
Process Noise Correlation Time Crrτ
Change in Process Noise ΔPN
Process Noise PN
Onset Measurement Noise Standard Deviation σo
Patient Zero Arrival Time t0
Simulated Recovery Data SRD
Simulated Onset Data SOD
Patient Zero Arrival PZA
Recovered R
Recovery Measurement Noise Standard Deviation σr
Recovery Rate RR
Time to Onset ts
Susceptible S
Exposed E
Infected I
Infection Fatality Rate ifr
Time to Removal tr
Population P
Exposure ER
Onset OR
Death D
Death Rate DR
Susceptible Fraction SF
Indicated Attack Rate IAR
Simulated Death Rate Data SDD
Indicated Death Rate IDR
<Death D>
Effect of Perceived Risk on Attack Rate EPA
Perceived Death Rate PDR
Death Measurement Noise Standard Deviation σd
Time to Reduce Risk td
Transmission Intensity β
Time to Perceive Risk TPR
Sensitivity to Death α
Death Risk Diminishing Impact γ
Time to Perceive tp

FIGURE 4 | Simplified structure of the SEIRb model.

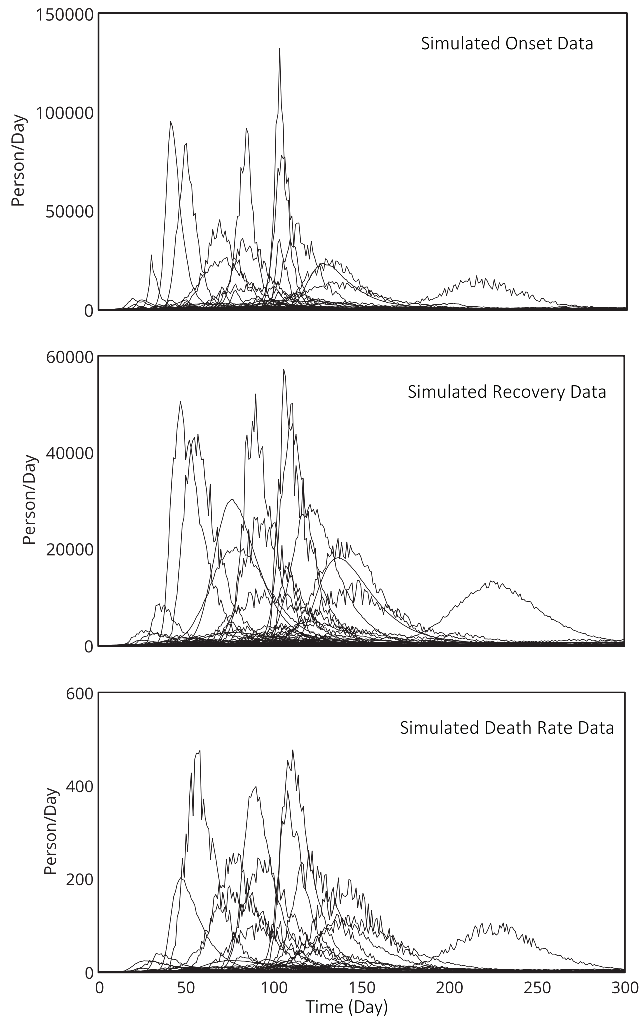


FIGURE 5 | SEIRb model behavior (200 simulations) using randomly drawn values from parameter priors.

TABLE 3 | Parameter priors of the SEIRb model that are estimated.

Name	Prior	Unit
Infection fatality rate	IFR = Uniform (0.003, 0.01)	Dimensionless
Sensitivity to death	α = Uniform (0.01, 100)	Day/Person
Transmission intensity	β = Uniform (0.1, 4)	1/Day
Death risk diminishing impact	γ = Uniform (0, 5)	Dimensionless
Time to perceive	t_p = Uniform (5, 100)	Day
Time to reduce risk	t_d = Uniform (10, 400)	Day
Patient zero arrival time	t_0 = Uniform (0, 100)	Day
Recovery measurement noise standard deviation	σ_r = Uniform (0, 0.3)	Dimensionless
Onset measurement noise standard deviation	σ_o = Uniform (0, 0.3)	Dimensionless
Death rate measurement noise standard deviation	σ_d = Uniform (0, 0.3)	Dimensionless
Process noise correlation time	Crr_τ = Uniform (2, 20)	Day
Process noise standard deviation	σ_p = Uniform (0, 0.3)	Dimensionless

for each model, we set a baseline (largely aligned with default parameters of BayesFlow) and assess how deviations in each parameter from that baseline impact training performance. Based on those results, we then specify an "optimized" hyperparameter setting for each model and conduct ABI using those hyperparameters and with different levels of simulation data informing inference. Below, we first report on the experiments with hyperparameters across both models and then share inference results for each model separately.

5.1 | Exploring Inference Hyperparameters

To have a better understanding of the role of hyperparameters in the inference performance (i.e., training and validation losses) and the training time, we conducted a series of experiments with both models before finalizing the setup for the main inference work. For these experiments, we defined baseline hyperparameter values based on the default settings provided by BayesFlow and our initial intuition of the required adjustments around those values. We chose the training parameters so that enough data is provided for inference to start working, but not too well, so there is room for improvement, and inference time is relatively short. Then, we varied the hyperparameters of inference around these baseline setups to observe the impact of individual assumptions on the training time and (validation sample) loss values for both Random Walk and SEIRb models.

From the insights we gained through these experiments, which will be discussed below, we came up with sets of hyperparameters that would provide the "Optimized" NN performance. Next, we used the "Optimized" hyperparameters with the same amount of data (as baseline) and an "Extended" version where more data (with optimized hyperparameters) provides the final outcome of inference. Table 4 summarizes the hyperparameters used in each scenario and some outcomes.

Variations in the hyperparameters result in changes in the (validation) loss value and training time, as shown in Figures 6 and 7. In general, more complex network structures (e.g., a higher number of convolutional layers or coupling layers) are more expressive and can offer better fit (smaller loss values; values below zero show getting to convergence, although the optimal value that signals perfect training will depend on the problem). However, they can increase the computational cost and may increase the risk of overfitting. In fact, convolutional layers do not offer much value

TABLE 4 | Hyperparameters used during neural network training for each model.

	Random Walk			SEIRb		
	Baseline	**Optimized**	**Extended**	**Baseline**	**Optimized**	**Extended**
Summary network						
Convolutional layers	2	4	4	3	4	4
Summary dimensions	10	20	20	30	40	40
LSTM units	128	128	128	128	128	128
Manual summary statistics	False	True	True	False	True	True
Inference network						
Coupling layers	4	6	6	6	8	8
Learning rate	0.0005	0.0010	0.0010	0.0010	0.0010	0.0010
Bidirectional	False	True	True	True	True	True
Training						
Batches	32	32	32	32	32	32
Epochs	20	20	10	20	20	10
Rounds	5	5	20	5	5	10
Sims per round	1024	1024	8192	8192	8192	20,000
Loss	−4.368	−4.148	−5.307	4.715	1.726	−4.574
Wall Time	5 min 28 s	6 min 4 s	4 h 19 min	1 h 5 min	1 h 12 min	5 h 40 min

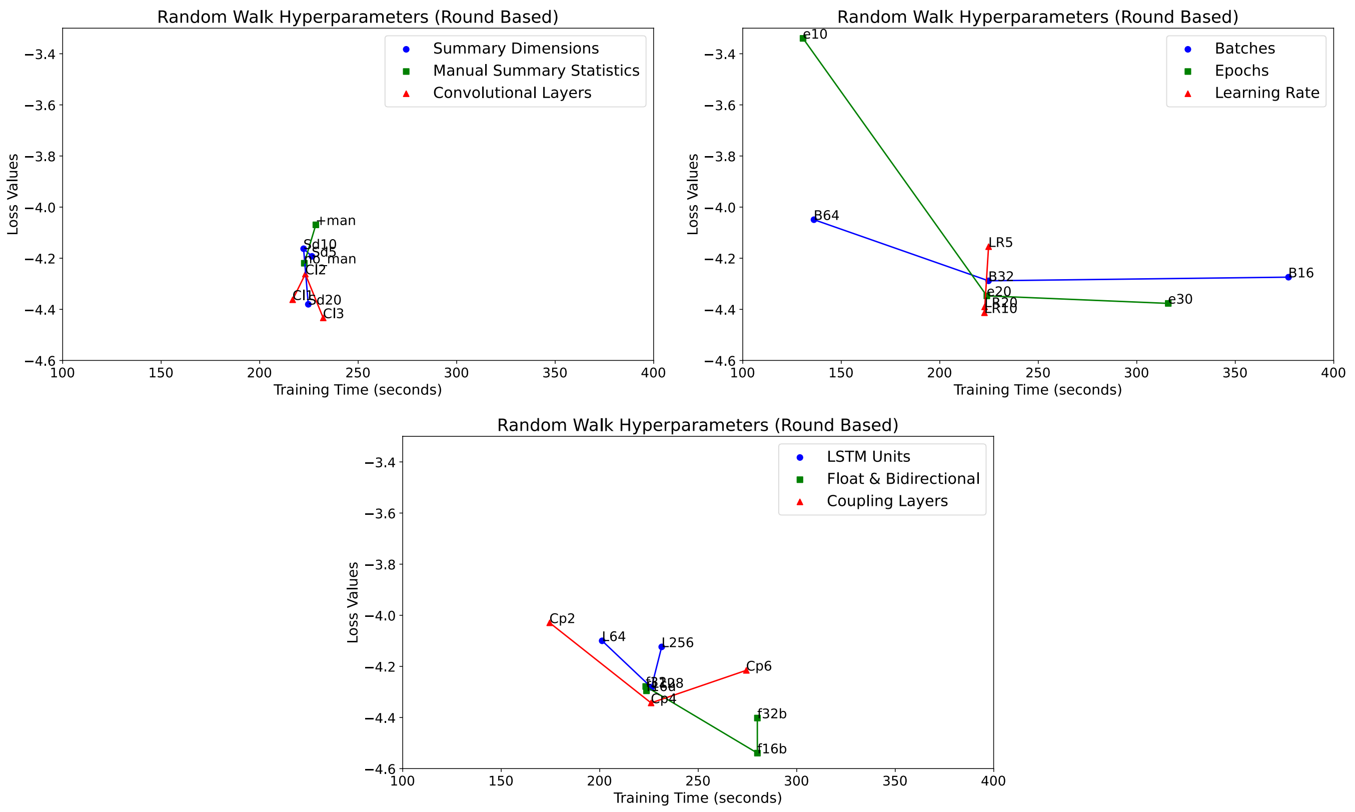


FIGURE 6 | Impact of hyperparameters on validation loss and training time for the Random Walk model. Labels denote specific configurations with their corresponding values: Sd5, Sd10, Sd20 for Summary Dimensions; +man (with manual statistics), no_man (without manual statistics); Cl1, Cl2, Cl3 for Convolutional Layers; B16, B32, B64 for Batches; e10, e20, e30 for Epochs; LR5, LR10, LR20 for Learning Rate (LR10 means: LR = 0.0010); L64, L128, L256 for LSTM Units; f16u, f32u (unidirectional), f16b, f32b (bidirectional) for Float; CP2, CP4, CP6 for Coupling Layers.

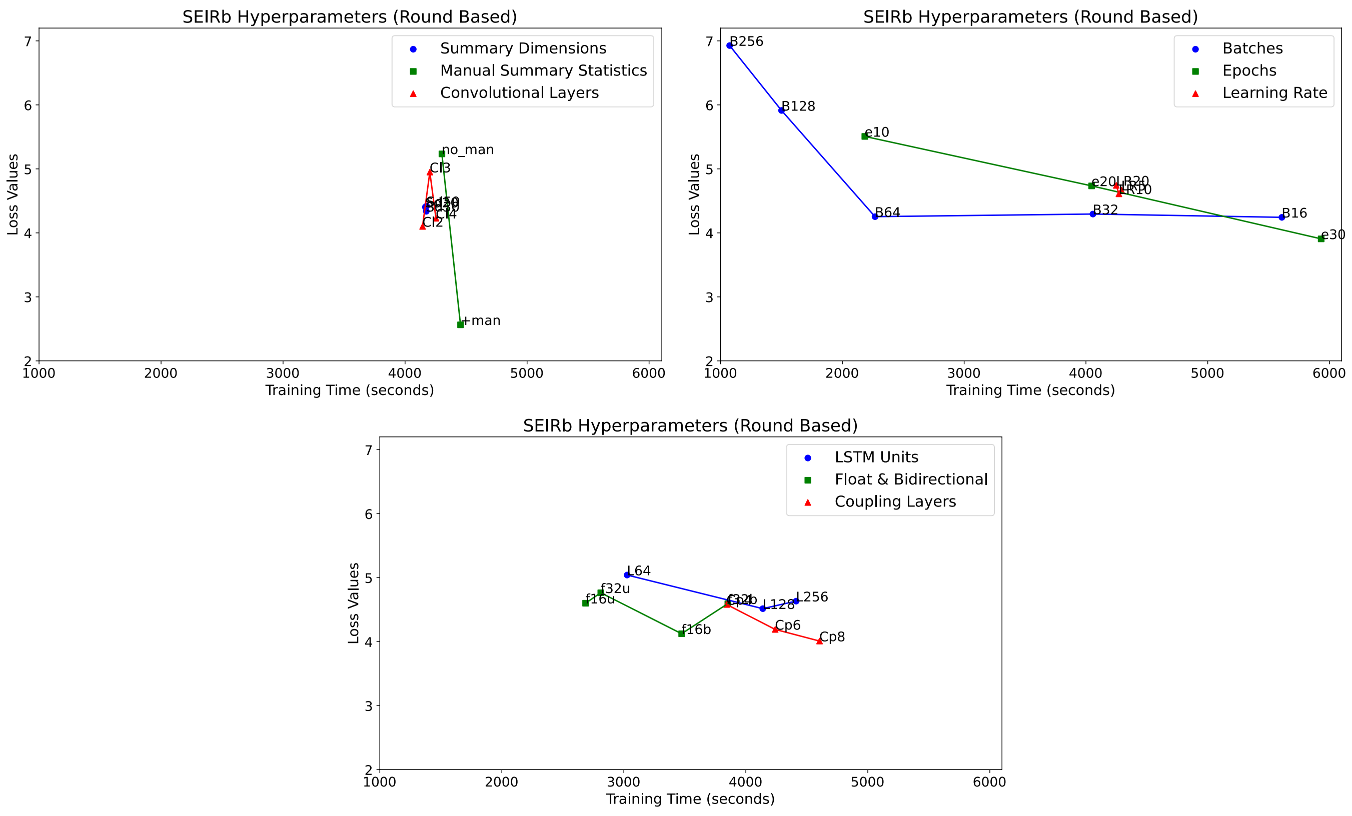


FIGURE 7 | Impact of hyperparameters on validation loss and training time for the SEIRb model. Labels denote specific configurations with their corresponding values: Sd20, Sd30, Sd50 for Summary Dimensions; +man (with manual statistics), no_man (without manual statistics); Cl2, Cl3, Cl4 for Convolutional Layers; B16, B32, B64, B128, B256 for Batches; e10, e20, e30 for Epochs; LR5, LR10, LR20 for Learning Rate (LR10 means: LR = 0.0010); L64, L128, L256 for LSTM Units; f16u, f32u (unidirectional), f16b, f32b (bidirectional) for Float; CP4, CP6, CP8 for Coupling Layers.

in our settings, with the LSTM part of the summary network having the bigger impact. We find value in using bidirectional LSTM summary networks, so we adopt that into our optimized settings and the baseline for SEIRb. Inference coupling layers and LSTM units are the largest contributors to NN training time (and bidirectional doubles the size of the LSTM network; so, it also impacts training time). Using float16 number formats offers slight savings but not enough to be worthwhile in our setup (given additional work to ensure robustness to smaller numerical precision).

Furthermore, keeping total data constant, more simulations per batch can significantly reduce the training time but at the expense of worse loss values (because the NN is updated fewer times). However, one could increase the number of epochs with larger batch sizes to get to a similar training time, making it challenging to find the optimal batch size. Nevertheless, the general recommendation is to avoid higher batch sizes to prevent memory allocation issues and use powers of 2 to utilize computational resources more efficiently. So, we find values of 32 and 64 to be best for our experiments and models. In addition, a greater number of epochs improves fit. However, they linearly increase the training time and become less valuable as the model starts to overfit after extracting the generalizable information in a dataset. We find learning rates only modestly impactful, and a value of 0.0010 is a good choice in our experiments.

More summary statistics could add some value up to a point (e.g., around 2–4 times the number of parameters) but become ineffective and potentially problematic if going much beyond that. To define manual summary statistics, we first calculate, for each data series, the residual between the data and a Savitzky-Golay second-order fitted line to the data. We then use the 0, 3, and 10 period lagged covariance matrix of these residuals across different data series. For example, in a Random Walk model that includes residual variance, 1-period lagged autocorrelation of residual, and 10-period lagged autocorrelation. For SEIRb, the covariances across residuals for different data series are added for different lags. We note that using manual summary statistics helps speed up the early identification of noise parameters but does not provide a longer-term benefit to the Random Walk model because it is already performing rather well by the end of training in the baseline. In contrast, manual summary statistics result in significant improvements in the SEIRb model (which is far from fully tuned in the baseline). Therefore, we adopt manual summary statistics for the final NN training. In addition, we were mindful of overfitting caused by using the same data multiple times (in multiple rounds and epochs), so in the extended training for the SEIRb model, while we increased the number of rounds too, we simultaneously decreased the number of epochs to avoid overfitting.

5.2 | Random Walk Inference Results

Here, we focus on reporting inference results from the "extended" run of the Random Walk model, though to provide better intuition on the impact of more/less training and data, we also compare that with the “optimized” run. To start, let us consider how an ABI inference is often utilized in practice, that is when we want to estimate model parameters based on one (or more) “empirical” datasets. The first thing to note is that ABI does NOT need the empirical dataset to build inference NNs! All it needs is the generative engine (the simulation model, the observable outcomes, and the priors) and settings for the training of NNs. The inference is completed by simulating the system dynamics model using different draws of the priors and training the NNs so that they identify reasonable summary statistics and are able to generate the posteriors for parameters given any simulated dataset. The main output of ABI is the trained summary and inference neural networks (e.g., for “optimized” and “extended” cases) and the “empirical” datasets play no role up to this point. When those NNs are trained, they could be used for inference on any “empirical” (or synthetic) dataset using the generative engine at hand.

Let us consider one concrete example of such a dataset for Random Walk. Figure 8a shows the time trajectory of a single dataset for the Random Walk model (in black dots connected with dashed lines). The dataset is indeed generated by the Random Walk model (i.e., is synthetic) and as such we know the true values of parameters and are confident that the model structure is “correct.” A real “empirical” dataset would have a similar data structure, but we would not know about the true parameter values. Now, we can input this dataset into the trained NNs and ask for samples of posteriors for model parameters. Figure 8b shows a sample of 500 draws from the posterior coming out of ‘extended’ inference in marginal and bi-variate plots. Black dots/dashed black lines point to the ground truth for each parameter. The diagram demonstrates that ground truth falls within the inferred posteriors. Table 5 reports the summary of posteriors, including median, mean, maximum a posteriori (MAP), and 95%-CI and the ground truth of the parameters used for generating the single synthetic dataset, indicating that posteriors are well informed by data.

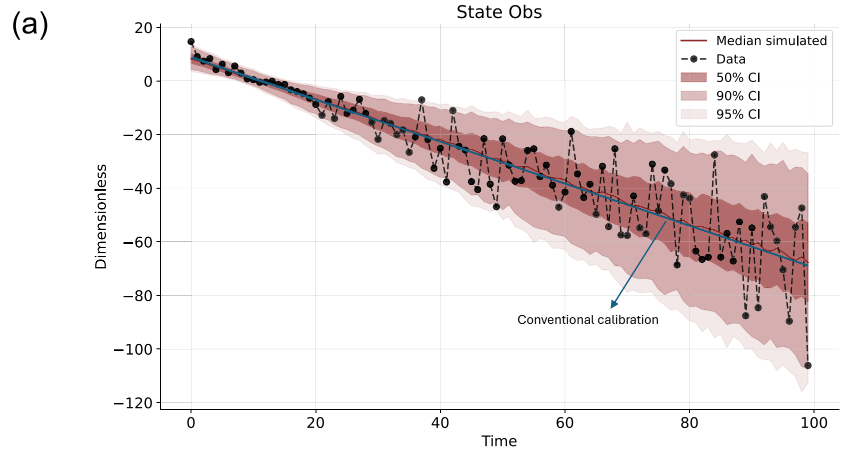
In other words, the algorithm has learned how to correctly identify the parameter values with reliable CIs, at least for this single dataset. In fact, one could take a sample of parameters from posterior, simulate the model again (with different noise seeds to generate independent random variations) and observe how the ensemble of the simulated outcomes vary over time. The relevant uncertainty intervals for such “posterior predictive check” are graphed in Figure 8a showing good correspondence with the data.

At this point, one might ask: how do these outcomes differ from the conventional calibration methods that most system dynamics modelers are familiar with? Figure 8a also includes the calibration trajectory obtained by minimizing the common least squared error between the simulation and data. That calibration offers estimates for two of the parameters: $d = -0.791$ and $S_0 = 9.530$. Comparing this outcome with the ones obtained through ABI reveals some key differences. First, the traditional calibration methods are deterministic, failing to explain the magnitude of variations around the mean and their ranges in data. Therefore, they offer no estimates for noise parameters (σ_p and σ_m). Second, unlike the ABI results, conventional calibration methods result in point estimates for parameter values and do not offer any information on the posterior distributions or credible intervals. So, we actually do not know how good those estimates are: could the drift value be -0.9? we cannot answer that in a simple calibration. These limitations highlight two key advantages of using an ABI.

A third issue arises when one asks whether the reasonable inference we obtained on a single dataset was merely by chance or the method can reliably recover the model parameters at scale. Traditional calibration is silent on this question, but ABI provides a systemic method to answer it. Specifically, we could conduct inference on many different (synthetic) datasets and compare the results against the ground truth for each. Figure 9a shows the results of such an exercise. Here, we do the inference 1000 times on different (synthetic) datasets (simulating the model with different parameters drawn from the priors and different noise streams). Then, for *each* synthetic dataset, we use the trained (in an “extended” scenario) NN to draw 1000 samples of the inferred *posteriors* for parameters. Note that the amortized nature of ABI makes these steps fast and easy: drawing 1000 posterior samples for the 1000 different synthetic datasets takes only a few seconds. With that data we graph the ground truth (x-axis) against the median of the posterior and the 10–90 percentile range on the y-axis for each estimated parameter. The closer the graph is to the 45° line with tighter 10–90 ranges, the tighter and more accurate the posteriors. The differences in accurate recovery of ground truth across parameters are notable. “Drift” is almost perfectly identified; S_0 and σ_M are also well identified, especially if the values for these two parameters are rather small. σ_P remains rather uncertain in many cases even after plenty of training (additional experiments show that more data and training do not reduce the uncertainty here). In other words, the variance in process noise is hard to pin down. The reason is that in this model, process noise is additive, whereas measurement noise is multiplicative. Therefore, the impact of measurement noise increases (to 10 s) as the stock values grow larger (in absolute), overwhelming any signal of the process noise (typically below 0.5 in each period) that could be detectable in the dataset, unless drift is close to zero, or measurement noise is very close to zero. Measurement noise is, however, easier to detect given how its impact scales with the size of the underlying stock.

The precision of the estimated credible intervals could also be assessed more formally. Figure 9b graphs the fraction of ground truth parameters that fall within different (centered) percentiles of posteriors. For example, a Y value of 0.52 at X of 0.5 means 52% of ground truth parameters have fallen within the centered 50% (i.e., 25–75 percentile interval) credible interval for the parameter. Perfect CIs fall on the 45° line, as they do in this example, with lines below 45° showing overconfident CIs and lines above pointing to over-conservatism of CIs.

Another way to visualize these effects is to consider two qualities of inferred parameters: how much the posterior has shrunk (*Posterior Contraction*: one minus the ratio of posterior standard deviation to that of prior) and if it is biased in comparison to ground truth. For each of the synthetic datasets,



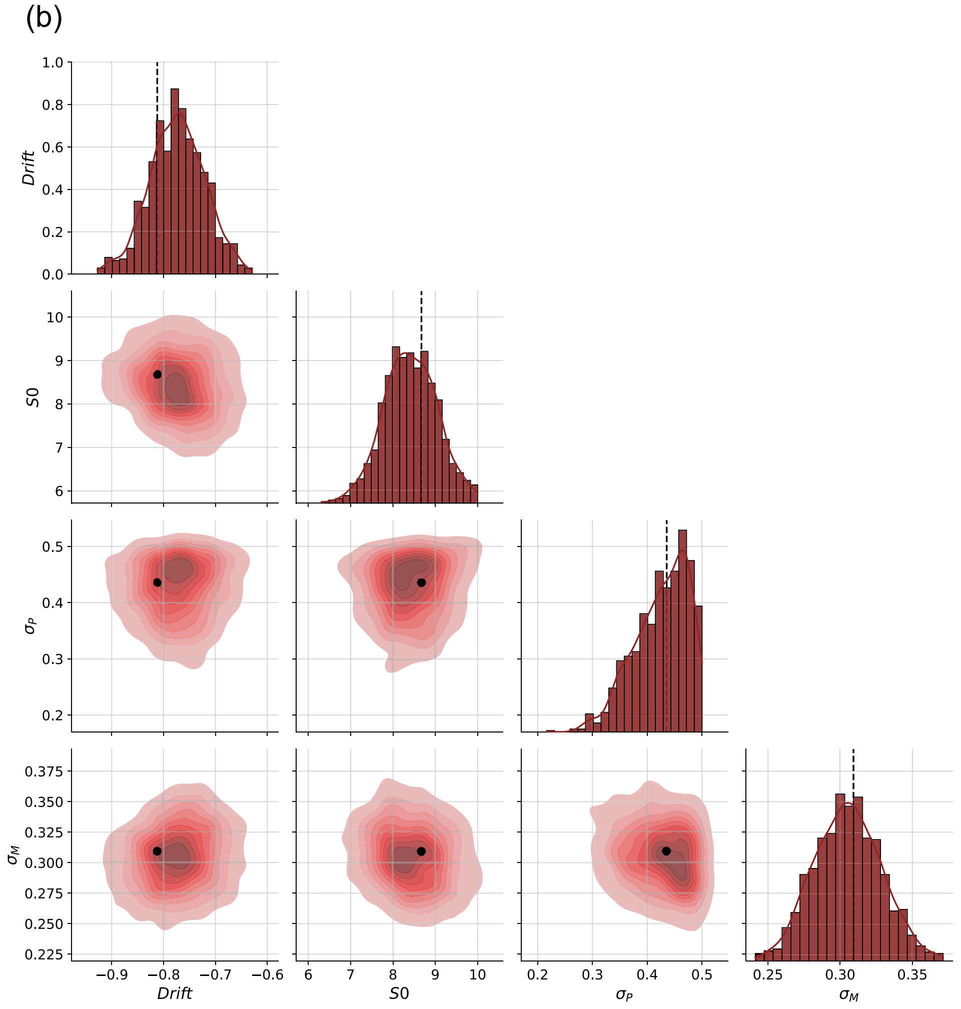


FIGURE 8 | (a) Posterior predictive check displaying the credible intervals derived from simulating the Random Walk model using 500 parameter samples drawn from the posterior distributions from the "extended" inference. The black dots connected by dashed lines represent the synthetic data (using ground truth parameters, and unknown noise streams), and (b) joint posterior distributions for 500 samples from the "extended" inference. Off-diagonal plots display the bivariate posterior distributions between parameter pairs, while the diagonal plots represent the marginal posterior distributions for each parameter. The vertical dashed lines (or black dots on the off-diagonal plots) indicate the ground truth of parameter values used to generate the synthetic data.

posterior contraction (x-axis) is graphed against the posterior Z-score (y-axis) in Figure 9c. The latter measures the distance between ground truth and the mean of the posterior sample for each estimation and divides that by the standard deviation of the posterior sample. Values centered around zero and not falling much outside of the [−2,2] range are usually desired and show limited bias. By this measure, all our parameter recoveries are unbiased, while the posterior contraction is high for all but σ_P, where it is smaller (and highly variable). Note that this observation does not point to a weakness of inference but the inherent complexity of estimating the process noise parameter here.

TABLE 5 | Posterior summaries and 95%-CIs for each model parameter inferred from the Random Walk model synthetic data generated using ground truth parameter values.

Parameter		Median	Mean	MAP	95%-CI	Ground truth	Unit
Drift	d	−0.773	−0.774	−0.771	[−0.883 to −0.669]	−0.812	1/Time
Initial state	S_0	8.406	8.417	8.288	[7.166 to 9.676]	8.673	Dimensionless
Process noise standard deviation	σ_p	0.435	0.426	0.464	[0.318 to 0.500]	0.436	1/Time
Measurement noise standard deviation	σ_m	0.305	0.305	0.305	[0.261 to 0.351]	0.309	Dimensionless

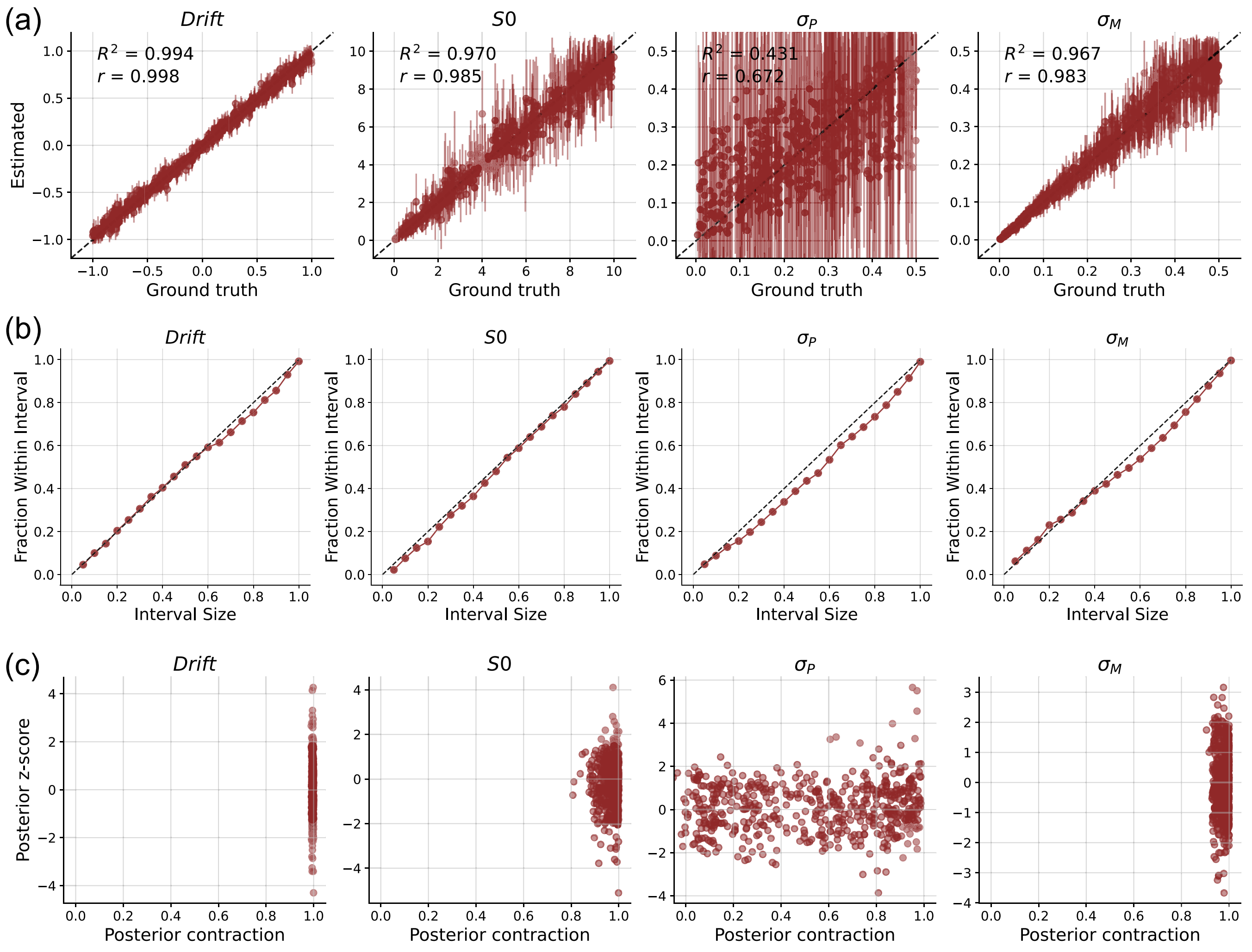


FIGURE 9 | Inference using 1000 different synthetic datasets from the Random Walk model, extended run, including the (a) estimated parameter against the ground truth, (b) fraction of ground truth parameters that fall within different (centered) percentiles of posteriors, and (c) posterior contraction against posterior Z-score.

ABI enabled the large-scale experiments above to be conducted in only a few seconds, with the majority of the time needed for simulating the synthetic datasets. These experiments provide one set of reassuring evidence about the reliability of inference. Another formal way to validate the inference procedure is to use Simulation Based Calibration (SBC; Note that "Calibration" in this case is used somewhat differently than the norm in the system dynamics community, pointing to fine-tuning of the inference procedure rather than identification of any single parameter). Starting with a sample of M prior draws, we will generate M datasets and N posterior draws per each dataset, comparing the distribution of M prior draws with the M*N samples of posteriors to test if they follow the same distribution. Formal tests are available for this purpose, and Figure 10 shows the ECDF rank test results for the Random Walk parameters. When the Rank graph exits the confidence bar (gray ovals for 95% in this case), it signals a potential mismatch between prior and posterior sample distributions. In this case, we see no such deviation (beyond luck) and thus can conclude that our inference method is working well and creating reliable posteriors for each parameter.

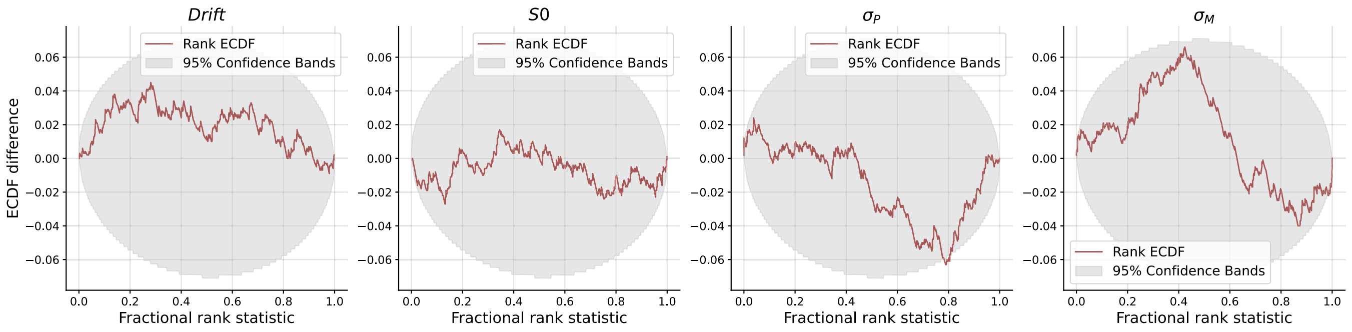


FIGURE 10 | Empirical cumulative distribution functions (ECDF) of rank statistics for Random Walk model, extended run.

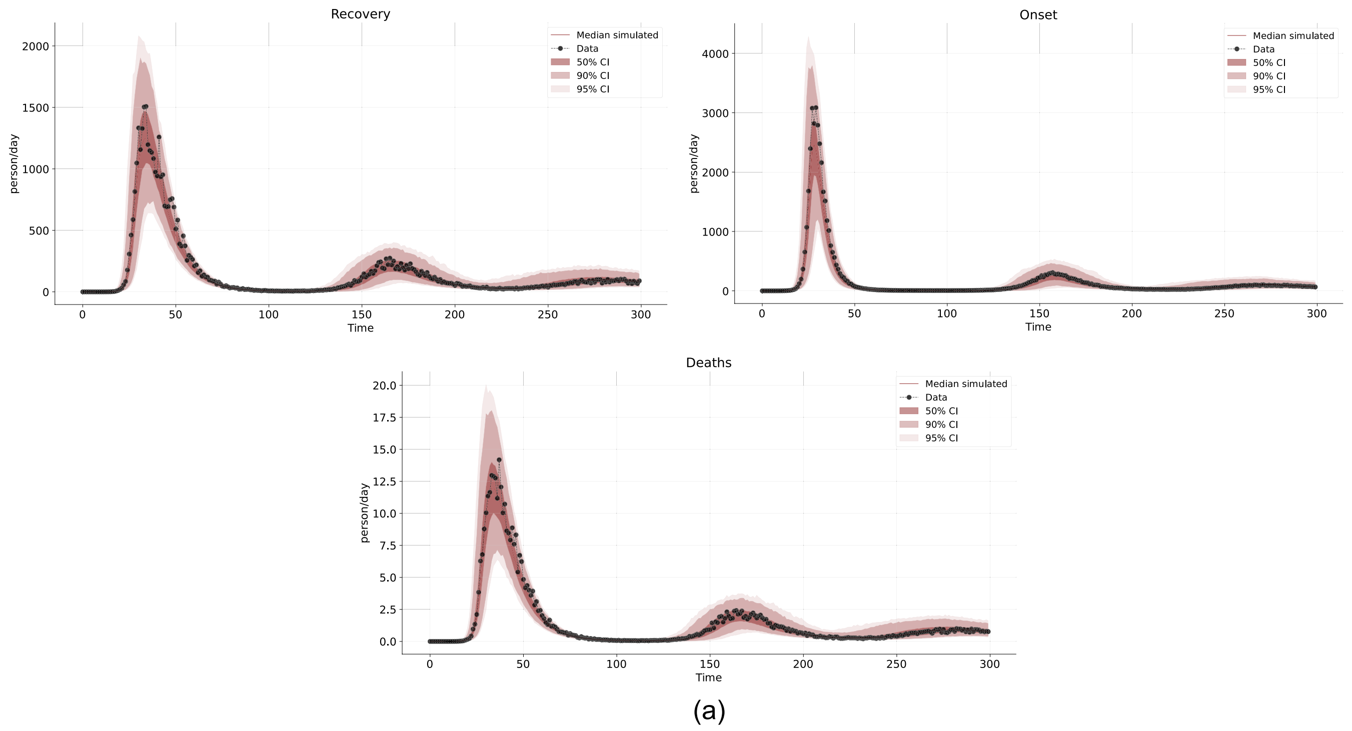


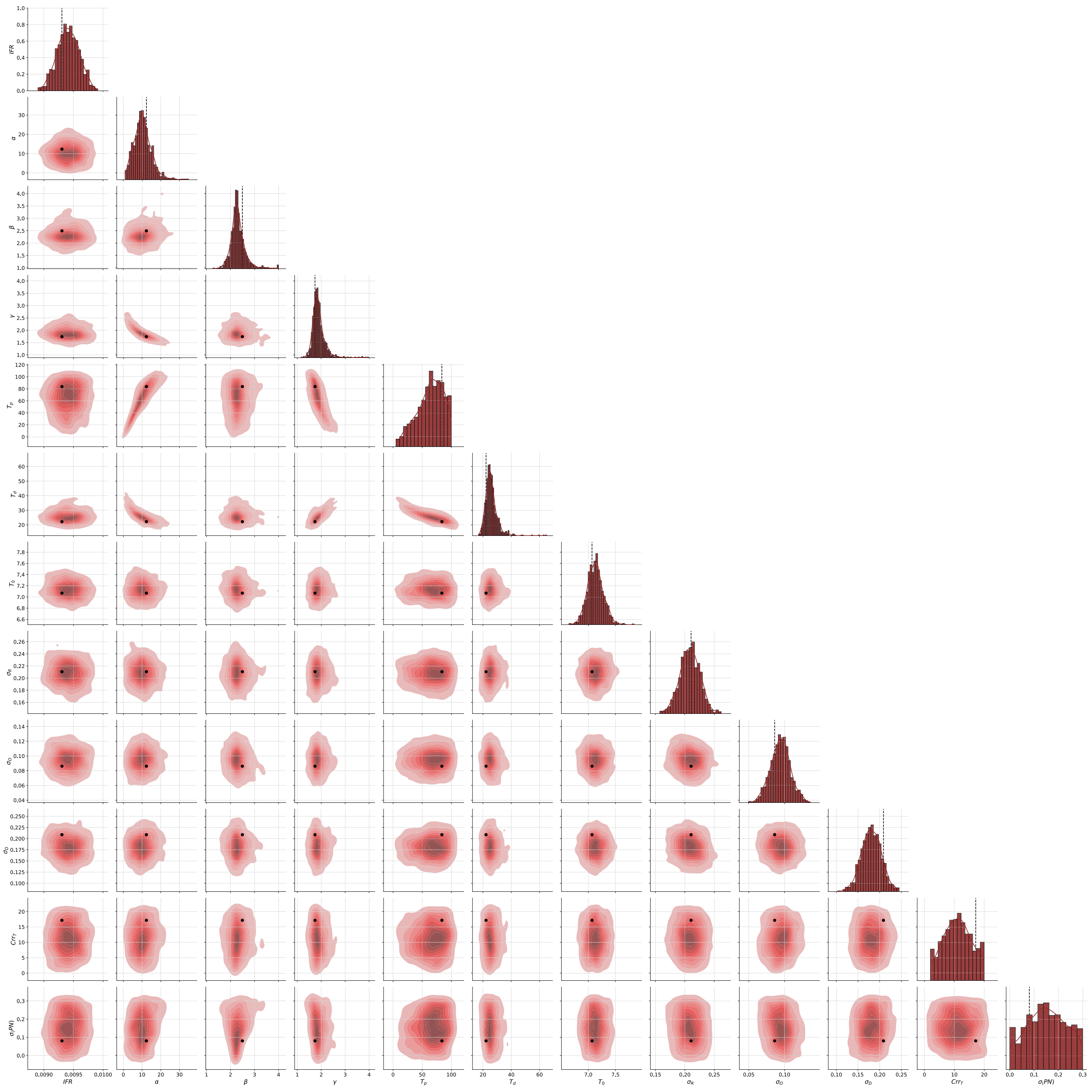
FIGURE 11 | (a) The synthetic data (black dots connected by dashed lines) used for inference enveloped by posterior predictive check displaying the credible intervals derived from simulating the SEIRb model using 500 parameter samples drawn from the posterior distributions from the "extended" inference, and (b) joint posterior distributions for 500 samples from the "extended" inference. Off-diagonal plots display the bivariate posterior distributions between parameter pairs, while the diagonal plots represent the marginal posterior distributions for each parameter. The vertical dashed lines (or black dots on the off-diagonal plots) indicate the ground truth of parameter values used to generate the synthetic data.

Overall, the results show that the inference procedure works well on the Random Walk model and offers reliable parameter estimates and CIs.

5.3 | SEIRb Inference Results

In reporting the results for SEIRb experiments, we follow the same flow as the Random Walk model. First, Figure 11 shows a single synthetic dataset (panel a, consisting of onset, recovery, and death data) and inferred posteriors and ground truth for the 12 parameters (panel b). The example dataset includes a large wave of the epidemic followed by smaller ones over a 300-day period. The synthetic dataset is generated using random draws from the prior values, and it is not used during the training of the neural network. Therefore, it is new unseen data for the NN.

Table 6 summarizes the posteriors, including median, mean, maximum a posteriori (MAP), and 95%-CI as well as the ground truth parameters used for generating the single synthetic dataset. A few observations are noteworthy. First, parameter posteriors are enveloping the ground truth very well (both in Figure 11 and Table 6). Second, posteriors for some parameters show significant interdependence, for example, higher values of α coincide with higher values of T_p in the posterior samples. The intuition is that the same level of behavioral response may point to higher responsiveness (α) combined with slower risk perception (higher T_p), or lower responsiveness combined with faster risk perception, an insight not previously noted in the analysis of the SEIRb model (Rahmandad, Xu, and Ghaffarzadegan 2022b). Reliable posteriors should reveal such interdependencies that matter conceptually. Third, the posterior for some parameters remains rather wide, most notably those related to parameters of



(b)

FIGURE 11 | (Continued)

process noise (Crr_T and σ_{PN}). The CIs for outcomes envelope the observed data closely and consistent with the implied uncertainties. So, overall, this single inference provides encouraging and consistent results. See Online Supporting Information for more details and another example.

Figure 12 reports on large-scale experiments for validating the inference framework for SEIRb more generally. The spread in recovered parameters (and their 10-90 CIs), as well as posterior contraction graphs, suggest reliable inference for IFR, T_0, γ, β, and measurement noise standard deviations (σ_R, σ_O, σ_D), weaker contraction for α, T_p (due to their collinearity; otherwise individually each would have been well identified) and T_d, very weak contraction for process noise standard deviation (σ_{PN}) and almost no contraction for process noise correlation time. These graphs do not establish if the parameters are efficiently identified (and thus not any better identifiable given this type of observed data) or that better identification can be achieved with more data (and/or training). To assess these possibilities, one needs to run more training and see if contraction scores improve at all.

It is encouraging to see that Z-scores show no systematic bias in recovered parameters, and we do not see many notable outliers. The CI precision plots are mostly on the 45° line, suggesting reliable CIs but with some deviations for T_0 and potentially IFR and Crr_T. In all those cases, the CIs are a bit too tight, a

TABLE 6 | Posterior summaries and 95%-CIs for each model parameter inferred from the SEIRb synthetic data generated using ground truth parameter values.

Parameter		Median	Mean	MAP	95%-CI	Ground truth	Unit
Infection fatality rate	IFR	0.009	0.009	0.009	[0.009–0.010]	0.009	Dimensionless
Sensitivity to death	α	10.029	10.362	9.893	[2.692–21.103]	12.327	Day/Person
Transmission intensity	β	2.278	2.319	2.255	[1.802–3.111]	2.495	1/Day
Death risk diminishing impact	γ	1.842	1.892	1.814	[1.524–2.533]	1.74	Dimensionless
Time to perceive	t_p	66.869	64.226	68.539	[16.951–100.000]	83.774	Day
Time to reduce risk	t_d	25.228	25.926	24.842	[19.789–36.290]	22.211	Day
Patient zero arrival time	t_0	7.129	7.131	7.138	[6.855–7.424]	7.068	Day
Recovery measurement noise standard deviation	σ_r	0.209	0.209	0.209	[0.175–0.241]	0.211	Dimensionless
Onset measurement noise standard deviation	σ_o	0.094	0.094	0.096	[0.067–0.122]	0.086	Dimensionless
Death rate measurement noise standard deviation	σ_d	0.181	0.181	0.179	[0.135–0.227]	0.209	Dimensionless
Process noise correlation time	Crr_τ	11.100	11.083	11.258	[2.000–20.000]	17.165	Day
Process noise standard deviation	σ_p	0.151	0.153	0.143	[0.000–0.300]	0.081	Dimensionless

potential concern as such bias may make the analyst overconfident. Those concerns are reinforced by the ECDF graphs in Figure 13. Again, while for most parameters, the test finds no difference in prior and sampled posterior distributions, for a few, it does. Those include the ones with imprecise CIs but also σ_D. Curiously, these parameters are the ones with very good contraction; that is, the model is identifying the ground truth very closely but is somehow failing in correctly inferring the shape of the posterior distribution, at least in some datasets.

While poor identification of Crr_T may explain the SBC divergence for that outcome, results for T_0 and IFR may, in fact, be pointing to a more general challenge in ABI. If the data very accurately identifies some parameter (such as these two), the posterior will be very tightly sampled (close to the ground truth). However, during the training of the inference and summary networks, the training samples are uniformly sampled from the *prior*, and thus, any small parameter region may get very few samples. For example, if the ground truth for T_0 is 7.1 (the neighborhood for our "empirical" dataset in Figure 11), the tight posterior will only be informed by samples falling between 6.7 and 7.5. Yet those samples are few (~1% of the total). The problem could be even more challenging. The higher the dimensionality of the parameter space, the more acute this problem can get because the curvature of the posterior for one parameter may depend on the values of other parameters. In those cases, the relevant combinations observed in training data shrink fast (with parameter dimensionality), reducing the accuracy of inferred posteriors. Interestingly, this challenge becomes more acute when the parameters are best identified by the available data. Overcoming this challenge may require a lot more training or giving up ABI in favor of sequential NPE.

Overall, the method is largely effective for the SEIRb example with unbiased posteriors that extract a lot of the information from the data and largely (but not fully) reliable posteriors. The problems in posteriors are most acute for the best-identified parameters with very sharp curvature in their posteriors not fully calibrated with the limited data in the relevant region, as well as the worst identified parameters.

5.4 | Impact of Training Budget

The results above focused on the "extended" runs, which include significantly more data than baseline or optimized setups. In Figure 14, we show the recovery and SBC plots for Random Walk from the optimized run to see how well inference performs with 32 times less data (1024×5 vs. 8096×20) and 56 times less training (given more training per dataset) than the extended case, which brings down the wall time for inference from 5h to 5min. The results are rather encouraging. Only the contraction for σ_p is notably weaker with posterior problems identified for that parameter, and perhaps for S_0. In short, the Random Walk problem is easy, and in fact, we can get comparable performance as the optimized scenario with even less training and data, completing a reasonable inference in less than 2min. That is partly made feasible using manual summary statistics that are especially helpful for identifying the σ_M quickly. Those summary statistics are specifically focused on characteristics of residuals and, as such, are especially informative about features of measurement noise that take more time to detect through automated summary statistics.

A similar comparison for SEIRb reinforces the same basic findings. With an order of magnitude less data/training, we can still identify most model parameters fairly well, with SBC outcomes that suggest the results are reliable (Figure 15). However, for some parameters (e.g., IFR), the CIs are much wider in the "optimized" compared to the "extended" scenario, even though the SBC plots hint at no problem in inference. This observation

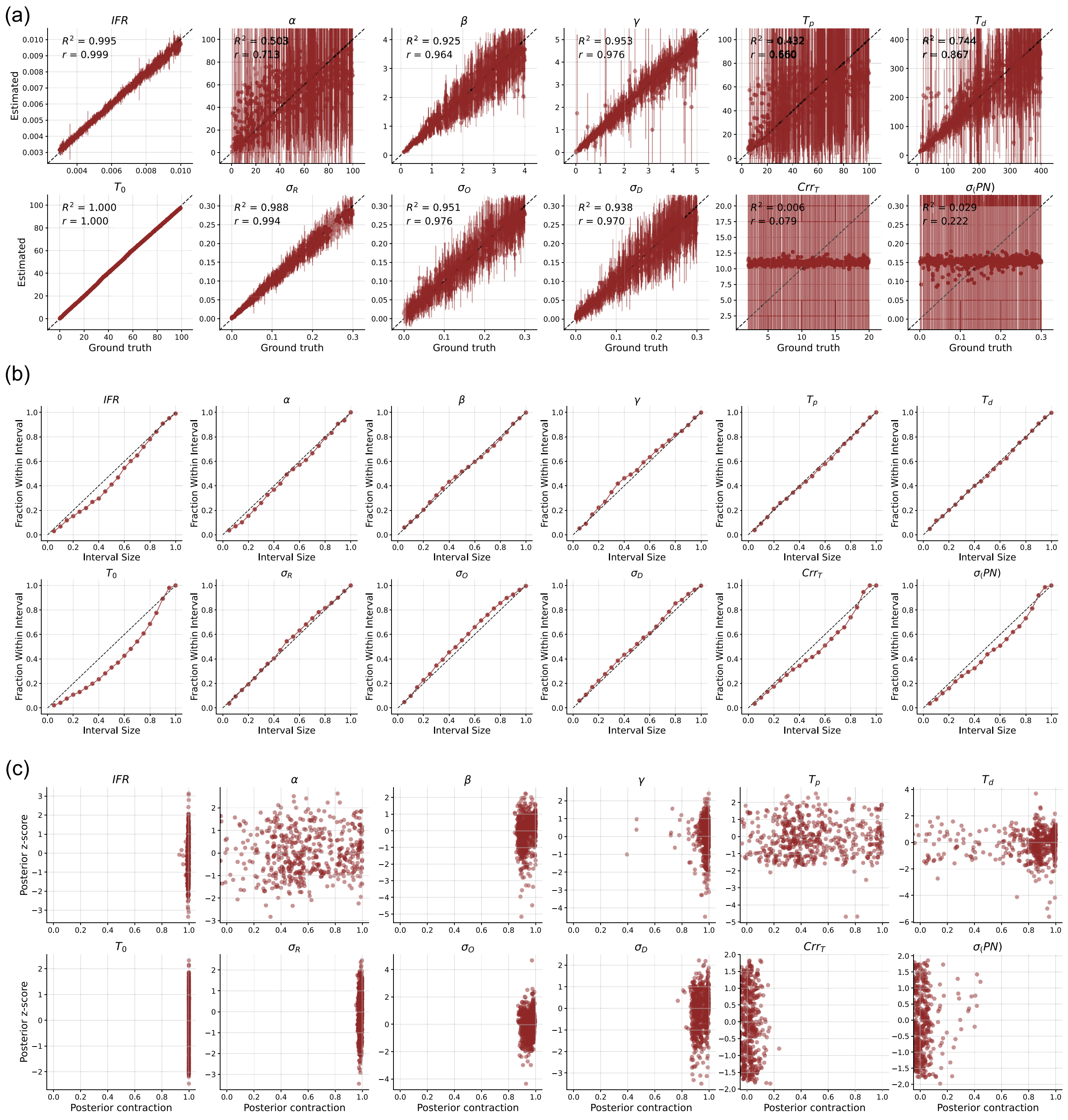


FIGURE 12 | Inference using 1000 different synthetic datasets from the SEIRb model, including the (a) estimated parameter against the ground truth, (b) fraction of ground truth parameters that fall within different (centered) percentiles of posteriors, and (c) posterior contraction against posterior Z-score.

suggests that the SBC method does not fully reveal the efficiency of inference and may confirm the reliability of a method with wider (than optimal) posteriors (but ones that show no bias in one direction or another). Additional explorations showed that, again, the manual summary statistics were especially helpful for identifying the measurement noise parameters. Payoff values (see Table 4) in baseline and optimized runs also point to room for improvement, as best payoffs would go below zero and are often smaller with larger numbers of parameters. Nevertheless, the trained networks start to offer useful inference outcomes for SEIRb with just 40 k simulations and an hour of training.

6 | Discussion

In this paper, we introduced the emerging neural network-based estimation methods for system dynamics practitioners. Just as people can do an impressive job of hand calibration by

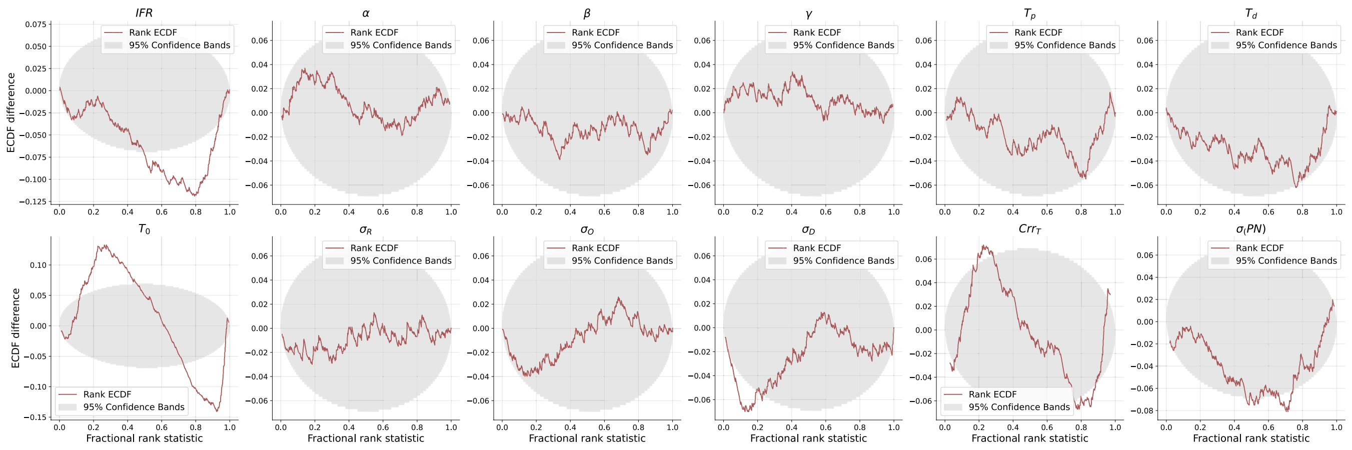


FIGURE 13 | Empirical cumulative distribution functions (ECDF) of rank statistics for SEIRb model.

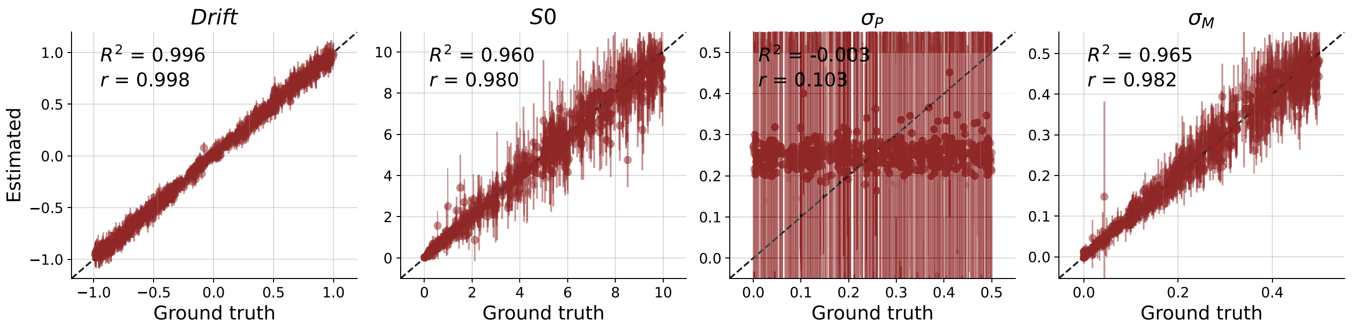
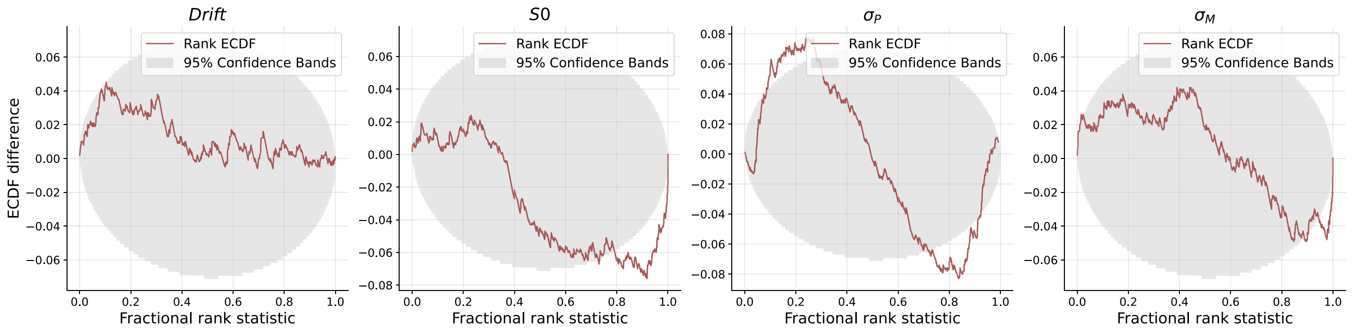



FIGURE 14 | Recovery and SBC plots for Random Walk from the optimized run.

observing a few simulations (Lyneis and Pugh 1996), neural networks can learn to associate different parameter combinations with different model behaviors to estimate model parameters and CIs. Focusing on ABI using NPE methods with established tools, we found that the parameters of simple to moderately complex system dynamics models could be effectively estimated in an amortized fashion. Using standard validation methods spanning SBC, recovery plots, CI inclusion fractions, posterior z-scores, and posterior contraction, we validate our inference procedure. The amortized nature of estimation enables straightforward validation of the method for any application and reduces computational costs when repetitions of estimation for different subjects, datasets, and so forth, are likely.

Much of this estimation machinery can be hidden from the typical user who can use the default hyperparameters and simple to use packages. Nevertheless, a few practical suggestions include:

- Having two to four times more summary statistics than parameters seems helpful. More summary statistics add little computational costs but may push models to overfit uninformative features of training data.
- Some parameters are identified with orders of magnitude less data and training than others. Identifying both process and measurement noise, especially the former, could be complex and require much data. So, one may be comfortable halting inference early if the identification of noise parameters was not of primary concern.
- Manual summary statistics are valuable and can speed up inference notably, especially for noise parameters. We found the covariance matrix for data residuals (compared to a de-noised mean tracker, e.g., Savitzky–Golay filter) against the lagged version of themselves and other data series as a helpful and generalizable summary statistic.

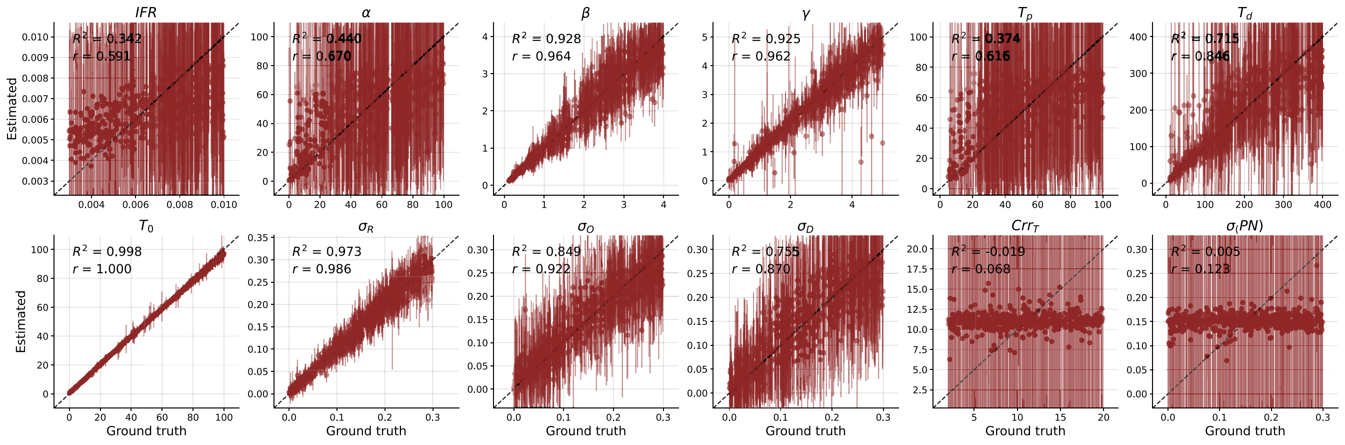
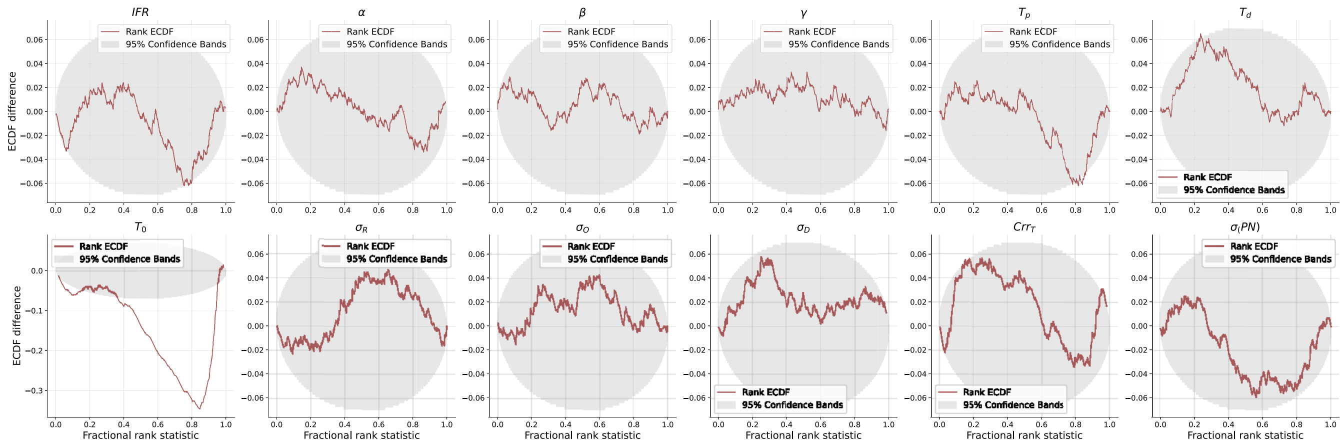



FIGURE 15 | Recovery and SBC plots for SEIRb from the optimized run.

- Bidirectional (rather than unidirectional) LSTM networks, by looking backward at data, are better at inferring initial conditions for stocks.
- Batch sizes of 16–64 and a learning rate of 0.001 seemed to be generally a good choice.
- Other hyperparameters may require problem-specific tuning: LSTM units (64 is a good starting point), number of inference layers (5–8 for simpler to more complex models), and convolutional layers for a sequential summary network (e.g., 3–4 layers, though their value was less salient).
- We found limited impact from normalizing parameters or log-transforming the data, but they may prove helpful when parameter ranges and model outputs are more varied.

This paper only provides an entry into the space, and several important questions remain. First, we do not know whether the estimated results are "efficient" (offering the tightest possible distribution of posteriors). SBC does not inform this question accurately. Therefore, testing efficiency on models with available likelihood-based estimation methods would be a valuable extension.

Scalability is another key question: Would this method work for typical system dynamics applications? Total computational costs in our ABI inference examples vary from a few minutes (for the Random Walk toy model) to a few hours for the SEIRb example. Better hardware can help. For example, we see 2–4 times savings moving from a desktop without GPUs to a similarly priced desktop with GPUs for NN training. Much of the computation is due to the NN training step: simulation time for sufficient sample sizes of $10^4 - 10^5$ takes a few seconds to complete when parallelized and compiled. For rather complex system dynamics models, $10^6 - 10^7$ simulations may be feasible, even easy, suggesting that the computational bottleneck currently remains with the NN training side, though memory constraints can also become problematic with large offline datasets.

Overall, the current machinery would allow for estimating typical system dynamics models in an amortized fashion with around two dozen parameters; for example, see (Radev et al. 2021). If one foregoes ABI and instead uses sequential NPE or NRE methods, computational costs could be cut further by an order of magnitude for estimating a single dataset. Some of those methods are built into a different package, SBI. Using good manual summary statistics also speeds up inference notably because they allow the posterior network to be trained from the outset; designing generic informative summary stats for dynamic models is promising. Scalability is not just about the number of parameters but also relates to the (often unknown) shape of the posterior distribution. Creating a larger number of test models with different parameter counts, reference modes, and noise structures and conducting continuous benchmarking research on that set would inform the scalability problem more precisely.

Another fruitful area is the comparison and expansion of model appropriateness measures. Prediction quality informs one set of these tests from simple fit measures (e.g., Root Mean Square Error [RMSE] and Mean Absolute Percentage Error [MAPE]) to complexity-adjusted ones (e.g., Akaike and Bayesian Information Criterion [AIC and BIC]). Out-of-sample predictive power (e.g., K-fold validation) is also used extensively. Another promising direction is the application of summary statistics, simulated vs. empirical, to assess the quality of the model in light of the data at hand (Gretton et al. 2012; Schmitt et al. 2022). Such tests inform a model's external validity (how close the model structure is to the true data generating process) and whether it has room for improvement. That would be a key additional input for an iterative model enhancement process. Nevertheless, even if we get an aggregate signal for weak external validity, the task of identifying the specific structural weaknesses in the model remains complex and should rely on engaging evidence from prior research, various stakeholders, and subject matter experts. Bridging the quantitative model testing approaches and these qualitative sources remains a key area for future contributions.

Another area for further exploration is the better integration of hierarchical models with the ABI machinery. For example, in modeling COVID-19, models for different countries may include the same "structure" but parameters that vary across countries and are potentially interrelated, creating a "hierarchical" architecture for model parameters. While ABI machinery already includes hierarchical estimation, this machinery may not be efficient if "driving data" (e.g., data inputs into the model) are different across different units. Efficient incorporation of driving data into inference so that inference costs scale sub-linearly with the number of units would be a valuable contribution.

Finally, transparency is critical for trust and adoption. It requires documentation of the modeling and estimation process, the assumptions made, the choice of neural network architectures, and the rationale for selecting specific hyperparameters. Sharing fully documented code and providing tested packages that bridge evolving estimation tools with system dynamics simulation software would prove valuable, and we hope this paper takes a first step in that direction.

More broadly, the field is rapidly evolving, with new methods and improvements emerging regularly. Packages that implement state-of-the-art methods (e.g., BayesFlow and Simulated-Based Inference, SBI) are critical for disseminating best practices, and bridging user communities with the evolving tools requires continued tool and bridge building. Our work takes a small step in that direction, but much more research is needed. The emerging tools can already solve some of the thorny estimation challenges the system dynamics community has dealt with (or was forced to side-step), and future breakthroughs in this space may ultimately 'solve' the estimation problem. This field thus offers exciting opportunities for researchers dedicated to refining and advancing the integration of quantitative data with nuanced and rich models system dynamics is best known for.

Acknowledgments

We thank ISDC 2024 conference reviewers for thoughtful feedback. We appreciate the computational resources provided by Università della Svizzera italiana for conducting some of the analyses in this paper.

Endnotes

[1] A notable exception to this statement is the use of Variational Inference (Blei, Kucukelbir, and McAuliffe 2017) methods which scale well but approximate the posterior distribution using a proxy distribution (e.g., Gaussian) and thus may not be appropriate for more complex posterior shapes.

[2] Perceived risk is regulated by the measured (rather than true) deaths. Thus, it is technically more precise to consider SDD to be impacted by a process noise because it feeds back into model dynamics, yet we use the more intuitive "measurement" label.

References

Andrade, J., and J. Duggan. 2021. "A Bayesian Approach to Calibrate System Dynamics Models Using Hamiltonian Monte Carlo." *System Dynamics Review* 37, no. 4: 283–309. https://doi.org/10.1002/sdr.1693.

Ardizzone, L., J. Kruse, C. Lüth, N. Bracher, C. Rother, and U. Köthe. 2021. "Conditional Invertible Neural Networks for Diverse Image-to-Image Translation." In *Pattern Recognition*, edited by Z. Akata, A. Geiger, and T. Sattler, 373–387. Cham: Springer International Publishing.

Arulampalam, M. S., S. Maskell, N. Gordon, and T. Clapp. 2002. "A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking." *IEEE Transactions on Signal Processing* 50, no. 2: 174–188. https://doi.org/10.1109/78.978374.

Blei, D. M., A. Kucukelbir, and J. D. McAuliffe. 2017. "Variational Inference: A Review for Statisticians." *Journal of the American Statistical Association* 112, no. 518: 859–877. https://doi.org/10.1080/01621459.2017.1285773.

Blei, D. M., and P. Smyth. 2017. "Science and Data Science." *Proceedings of the National Academy of Sciences* 114, no. 33: 8689–8692. https://doi.org/10.1073/pnas.1702076114.

Box, G. E., and G. M. Jenkins. 1976. *Time Series Analysis: Forecasting and Control* (Rev. ed.). San Francisco: Holden-Day.

Box, G. E. P., W. G. Hunter, and J. S. Hunter. 1978. *Statistics for Experimenters: An Introduction to Design, Data Analysis, and Model Building*. New York: Wiley.

Casella, G., and R. L. Berger. 1990. *Statistical Inference*. Belmont, CA: Duxbury Press.

Cranmer, K., J. Brehmer, and G. Louppe. 2020. "The Frontier of Simulation-Based Inference." *Proceedings of the National Academy of Sciences* 117, no. 48: 30055–30062. https://doi.org/10.1073/pnas.1912789117.

De Melo, C. M., A. Torralba, L. Guibas, J. DiCarlo, R. Chellappa, and J. Hodgins. 2022. "Next-Generation Deep Learning Based on Simulators and Synthetic Data." *Trends in Cognitive Sciences* 26, no. 2: 174–187. https://doi.org/10.1016/j.tics.2021.11.008.

Drovandi, C., and D. T. Frazier. 2022. "A Comparison of Likelihood-Free Methods With and Without Summary Statistics." *Statistics and Computing* 32, no. 3: 42. https://doi.org/10.1007/s11222-022-10092-4.

Durkan, C., A. Bekasov, I. Murray, and G. Papamakarios. 2019. "Neural Spline Flows." In *Advances in Neural Information Processing Systems*, edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Vol. 32. Vancouver, Canada: Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2019/file/7ac71d433f282034e088473244df8c02-Paper.pdf.

Eberlein, R. 2015. “Working With Noisy Data: Kalman Filtering and State Resetting.” In *Analytical Methods for Dynamic Modelers*. Cambridge, MA: MIT Press.

Forrester, J. W. 1961. *Industrial Dynamics*. Cambridge, MA: MIT Press.

Forrester, J. W. 1987. “Lessons From System Dynamics Modeling.” *System Dynamics Review* 3, no. 2: 136–149. https://doi.org/10.1002/sdr.4260030205.

Gelman, A., J. B. Carlin, H. S. Stern, and D. B. Rubin. 1995. *Bayesian Data Analysis*. New York: Chapman and Hall/CRC.

Gelman, A., A. Vehtari, D. Simpson, et al. 2020. “Bayesian Workflow.”

Gershman, S., and N. Goodman. 2014. “Amortized Inference in Probabilistic Reasoning.” In *Proceedings of the Annual Meeting of the Cognitive Science Society*.

Gill, J. 2002. *Bayesian Methods: A Social and Behavioral Sciences Approach*. Boca Raton, FL: Chapman and Hall/CRC.

Greenberg, D., M. Nonnenmacher, and J. Macke. 2019. “Automatic Posterior Transformation for Likelihood-Free Inference.” In *Proceedings of the 36th International Conference on Machine Learning*, 2404–2414: PMLR.

Gretton, A., K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola. 2012. “A Kernel Two-Sample Test.” *Journal of Machine Learning Research* 13, no. 1: 723–773.

Hansen, L. P. 1982. “Large Sample Properties of Generalized Method of Moments Estimators.” *Econometrica* 50, no. 4: 1029–1054. https://doi.org/10.2307/1912775.

Hermans, J., V. Begy, and G. Louppe. 2020. “Likelihood-Free MCMC With Amortized Approximate Ratio Estimators.” In *Proceedings of the 37th International Conference on Machine Learning*, 4239–4248: PMLR.

Homer, J. B. 1996. “Why We Iterate: Scientific Modeling in Theory and Practice.” *System Dynamics Review* 12, no. 1: 1–19. https://doi.org/10.1002/(SICI)1099-1727(199621)12:1<1::AID-SDR93>3.0.CO;2-P.

Hosseinichimeh, N., H. Rahmandad, M. S. Jalali, and A. K. Wittenborn. 2016. “Estimating the Parameters of System Dynamics Models Using Indirect Inference.” *System Dynamics Review* 32, no. 2: 156–180. https://doi.org/10.1002/sdr.1558.

Jalali, M., H. Rahmandad, and H. Ghoddusi. 2015. “Using the Method of Simulated Moments for System Identification.” In *Analytical Methods for Dynamic Modelers*, 39–69. Cambridge, MA: MIT Press.

Kennedy, P. 2008. *A Guide to Econometrics*. Malden, MA: Blackwell Pub.

Kingma, D. P., and P. Dhariwal. 2018. “Glow: Generative Flow With Invertible 1×1 Convolutions.” In *Advances in Neural Information Processing Systems*, edited by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Vol. 31. Montréal, Canada: Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2018/file/d139db6a236200b21cc7f752979132d0-Paper.pdf.

Kingma, D. P., and M. Welling. 2022. “Auto-Encoding Variational Bayes.”

Li, T., H. Rahmandad, and J. Sterman. 2022. “Improving Parameter Estimation of Epidemic Models: Likelihood Functions and Kalman Filtering.”

Lueckmann, J.-M., J. Boelts, D. Greenberg, P. Goncalves, and J. Macke. 2021. “Benchmarking Simulation-Based Inference.” In *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics*, 343–351: PMLR.

Lueckmann, J.-M., P. J. Goncalves, G. Bassetto, K. Öcal, M. Nonnenmacher, and J. H. Macke. 2017. “Flexible Statistical Inference for Mechanistic Models of Neural Dynamics.” In *Advances in Neural Information Processing Systems*, edited by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Long Beach, CA: Curran Associates, Inc. https://proceedings.neurips.cc/paper_files/paper/2017/file/addfa9b7e234254d26e9c7f2af1005cb-Paper.pdf.

Lyneis, J. M., and A. L. Pugh. 1996. “Automated vs. Hand Calibration of System Dynamics Models: An Experiment With a Simple Project Model.” In *Proceedings of the 1996 International System Dynamics Conference*. MA: System Dynamics Society Cambridge.

Manski, C. F. 2013. “Public Policy in an Uncertain World: Analysis and Decisions.” In *Public Policy in an Uncertain World*. Cambridge, MA: Harvard University Press.

Marin, J.-M., P. Pudlo, C. P. Robert, and R. J. Ryder. 2012. “Approximate Bayesian Computational Methods.” *Statistics and Computing* 22, no. 6: 1167–1180. https://doi.org/10.1007/s11222-011-9288-2.

Nikolenko, S. I. 2021. *Synthetic Data for Deep Learning*. Cham: Springer International Publishing.

Oliva, R. 2003. “Model Calibration as a Testing Strategy for System Dynamics Models.” *European Journal of Operational Research* 151, no. 3: 552–568. https://doi.org/10.1016/S0377-2217(02)00622-7.

Papamakarios, G., E. Nalisnick, D. J. Rezende, S. Mohamed, and B. Lakshminarayanan. 2021. “Normalizing Flows for Probabilistic Modeling and Inference.” *Journal of Machine Learning Research* 22, no. 1: 57:2617–57:2680.

Popper, K. 1934. *The Logic of Scientific Discovery*. New York: Routledge.

Radev, S. T., F. Graw, S. Chen, et al. 2021. “OutbreakFlow: Model-Based Bayesian Inference of Disease Outbreak Dynamics With Invertible Neural Networks and Its Application to the COVID-19 Pandemics in Germany.” *PLoS Computational Biology* 17, no. 10: e1009472. https://doi.org/10.1371/journal.pcbi.1009472.

Radev, S. T., U. K. Mertens, A. Voss, L. Ardizzone, and U. Köthe. 2022. “BayesFlow: Learning Complex Stochastic Models With Invertible Neural Networks.” *IEEE Transactions on Neural Networks and Learning Systems* 33, no. 4: 1452–1466. https://doi.org/10.1109/TNNLS.2020.3042395.

Radev, S. T., M. Schmitt, L. Schumacher, et al. 2023. “BayesFlow: Amortized Bayesian Workflows With Neural Networks.”

Rahmandad, H., T. Y. Lim, and J. Sterman. 2021. “Behavioral Dynamics of COVID-19: Estimating Underreporting, Multiple Waves, and Adherence Fatigue Across 92 Nations.” *System Dynamics Review* 37, no. 1: 5–31. https://doi.org/10.1002/sdr.1673.

Rahmandad, H., and J. Sterman. 2022. “Quantifying the COVID-19 Endgame: Is a New Normal Within Reach?” *System Dynamics Review* 38, no. 4: 329–353. https://doi.org/10.1002/sdr.1715.

Rahmandad, H., R. Xu, and N. Ghaffarzadegan. 2022a. “A Missing Behavioural Feedback in COVID-19 Models Is the Key to Several Puzzles.” *BMJ Global Health* 7, no. 10: e010463. https://doi.org/10.1136/bmjgh-2022-010463.

Rahmandad, H., R. Xu, and N. Ghaffarzadegan. 2022b. “Enhancing Long-Term Forecasting: Learning From COVID-19 Models.” *PLoS Computational Biology* 18, no. 5: e1010100. https://doi.org/10.1371/journal.pcbi.1010100.

Raissi, M., P. Perdikaris, and G. E. Karniadakis. 2019. “Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations.” *Journal of Computational Physics* 378: 686–707. https://doi.org/10.1016/j.jcp.2018.10.045.

Säilynoja, T., P.-C. Bürkner, and A. Vehtari. 2022. “Graphical Test for Discrete Uniformity and Its Applications in Goodness-of-Fit Evaluation and Multiple Sample Comparison.” *Statistics and Computing* 32, no. 2: 32. https://doi.org/10.1007/s11222-022-10090-6.

Schmitt, M., P.-C. Bürkner, U. Köthe, and S. T. Radev. 2022. “Detecting Model Misspecification in Amortized Bayesian Inference With Neural Networks.”

Sterman, J. 2000. *Business Dynamics: Systems Thinking and Modeling for a Complex World*. Boston: McGraw-Hill Education.

Sterman, J. 2018. “System Dynamics at Sixty: The Path Forward.” *System Dynamics Review* 34, no. 1–2: 5–47. https://doi.org/10.1002/sdr.1601.

Tabak, E. G., and C. V. Turner. 2013. “A Family of Nonparametric Density Estimation Algorithms.” *Communications on Pure and Applied Mathematics* 66, no. 2: 145–164. https://doi.org/10.1002/cpa.21423.

Talts, S., M. Betancourt, D. Simpson, A. Vehtari, and A. Gelman. 2020. “Validating Bayesian Inference Algorithms With Simulation-Based Calibration.”

Tran, D., M. Dusenberry, M. van der Wilk, and D. Hafner. 2019. “Bayesian Layers: A Module for Neural Network Uncertainty.” In *Advances in Neural Information Processing Systems*. Vancouver, Canada: Curran Associates, Inc.

Varian, H. R. 2014. “Big Data: New Tricks for Econometrics.” *Journal of Economic Perspectives* 28, no. 2: 3–28. https://doi.org/10.1257/jep.28.2.3.

Vehtari, A., A. Gelman, and J. Gabry. 2017. “Practical Bayesian Model Evaluation Using Leave-One-Out Cross-Validation and WAIC.” *Statistics and Computing* 27, no. 5: 1413–1432. https://doi.org/10.1007/s11222-016-9696-4.

Supporting Information

Additional supporting information can be found online in the Supporting Information section.