Sequential Neural Likelihood: Fast Likelihood-free Inference with Autoregressive Flows

George Papamakarios
University of Edinburgh

David C. Sterratt
University of Edinburgh

Iain Murray
University of Edinburgh

Abstract

We present Sequential Neural Likelihood (SNL), a new method for Bayesian inference in simulator models, where the likelihood is intractable but simulating data from the model is possible. SNL trains an autoregressive flow on simulated data in order to learn a model of the likelihood in the region of high posterior density. A sequential training procedure guides simulations and reduces simulation cost by orders of magnitude. We show that SNL is more robust, more accurate and requires less tuning than related neural-based methods, and we discuss diagnostics for assessing calibration, convergence and goodness-of-fit.

1 Introduction

In many areas of science and engineering, a natural way to model a stochastic system of interest is as a *simulator*, which may be controlled by potentially interpretable parameters and can be run forward to generate data. Such simulator models lend themselves naturally to modelling mechanistic processes such as particle collisions in high energy physics [1, 70], universe evolution in cosmology [2, 66], stochastic dynamical systems in ecology and evolutionary biology [63, 64, 82, 83], macroeconomic models in econometrics [3, 25], and biological neural networks in neuroscience [47, 61, 71]. Simulators are a powerful and flexible modelling tool; they can take the form of a computer graphics engine [44] or a physics engine [84], and they can become part of probabilistic programs [14, 16, 36].

Due to the wide applicability of simulator models, several tasks of interest can be framed as inference of the parameters of a simulator from observed data. For instance, inferring the parameters of physical theories given particle collisions can be the means for discovering new physics [11], and inferring the scene parameters of a graphics engine given a natural image can be the basis for computer vision [65, 85]. Hence, the development of accurate and efficient general-purpose inference methods for simulator models can have a profound impact in scientific discovery and artificial intelligence.

Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS) 2019, Naha, Okinawa, Japan. PMLR: Volume 89. Copyright 2019 by the author(s).

The fundamental difficulty in inferring the parameters of a simulator given data is the unavailability of the likelihood function. In Bayesian inference, posterior beliefs about parameters $\boldsymbol{\theta}$ given data $\mathbf{x}$ can be obtained by multiplying the likelihood $p(\mathbf{x}\,|\,\boldsymbol{\theta})$ with prior beliefs $p(\boldsymbol{\theta})$ and normalizing. However, calculating the likelihood $p(\mathbf{x}\,|\,\boldsymbol{\theta})$ of a simulator model for given parameters $\boldsymbol{\theta}$ and data $\mathbf{x}$ is computationally infeasible in general, and hence traditional likelihood-based Bayesian methods, such as variational inference [81] or Markov Chain Monte Carlo [53], are not directly applicable.

To overcome this difficulty, several methods for *likelihood-free inference* have been developed, such as Approximate Bayesian Computation [45] and Synthetic Likelihood [83], that require only the ability to generate data from the simulator. Such methods simulate the model repeatedly, and use the simulated data to build estimates of the parameter posterior. In general, the accuracy of likelihood-free inference improves as the number of simulations increases, but so does the computational cost, especially if the simulator is expensive to run. The challenge in developing new likelihood-free inference methods is to achieve a good trade-off between estimation accuracy and simulation cost.

In this paper we present *Sequential Neural Likelihood (SNL)*, a new likelihood-free method for Bayesian inference of simulator models, based on state-of-the-art conditional neural density estimation with autoregressive flows. The main idea of SNL is to train a Masked Autoregressive Flow [59] on simulated data in order to estimate the conditional probability density of data given parameters, which then serves as an accurate model of the likelihood function. During training, a Markov Chain Monte Carlo sampler selects the next batch of simulations to run using the most up-to-date estimate of the likelihood function, leading to a reduction in the number of simulations of several orders of magnitude. To aid the practitioner in deploying SNL, we discuss practical issues such as implementation and tuning, diagnosing convergence, and assessing goodness-of-fit. Experimental results show that SNL is robust and well-calibrated, and that it outperforms existing state-of-the-art methods based on neural density estimation both in terms of posterior estimation accuracy and simulation cost.

2 Likelihood-free inference with neural density estimation

A simulator model is a computer program, which takes a vector of parameters $\boldsymbol{\theta}$, makes internal calls to a random number generator, and outputs a data vector $\mathbf{x}$. Implicitly, this procedure defines a conditional probability distribution $p(\mathbf{x} \,|\, \boldsymbol{\theta})$ which in general we cannot evaluate, but we can easily sample from by running the program. Given an observed data vector $\mathbf{x}_o$ and a prior distribution $p(\boldsymbol{\theta})$, we are interested in estimating the parameter posterior $p(\boldsymbol{\theta} \,|\, \mathbf{x}_o) \propto p(\mathbf{x}_o \,|\, \boldsymbol{\theta})\, p(\boldsymbol{\theta})$.

This paper focuses on an approach to likelihood-free inference that is based on neural density estimation. A *conditional neural density estimator* is a parametric model $q_{\boldsymbol{\phi}}$ (such as a neural network) controlled by a set of parameters $\boldsymbol{\phi}$, which takes a pair of datapoints $(\mathbf{u}, \mathbf{v})$ and outputs a conditional probability density $q_{\boldsymbol{\phi}}(\mathbf{u} \,|\, \mathbf{v})$. Given a set of training data $\{\mathbf{u}_n, \mathbf{v}_n\}_{1:N}$ that are independent and identically distributed according to a joint probability density $p(\mathbf{u}, \mathbf{v})$, we can train $q_{\boldsymbol{\phi}}$ by maximizing the total log probability $\sum_n \log q_{\boldsymbol{\phi}}(\mathbf{u}_n \,|\, \mathbf{v}_n)$ with respect to $\boldsymbol{\phi}$. With enough training data, and with a sufficiently flexible model, $q_{\boldsymbol{\phi}}(\mathbf{u} \,|\, \mathbf{v})$ will learn to approximate the conditional $p(\mathbf{u} \,|\, \mathbf{v})$.

We can use a neural density estimator $q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \,|\, \mathbf{x})$ that models the conditional of parameters given data to approximate the posterior $p(\boldsymbol{\theta} \,|\, \mathbf{x}_o)$ as follows. First, we obtain a set of samples $\{\boldsymbol{\theta}_n, \mathbf{x}_n\}_{1:N}$ from the joint distribution $p(\boldsymbol{\theta}, \mathbf{x})$, by $\boldsymbol{\theta}_n \sim p(\boldsymbol{\theta})$ and $\mathbf{x}_n \sim p(\mathbf{x} \,|\, \boldsymbol{\theta}_n)$ for $n = 1, \ldots, N$. Then, we train $q_{\boldsymbol{\phi}}$ using $\{\boldsymbol{\theta}_n, \mathbf{x}_n\}_{1:N}$ as training data in order to obtain a global approximation of $p(\boldsymbol{\theta} \,|\, \mathbf{x})$. Finally, $p(\boldsymbol{\theta} \,|\, \mathbf{x}_o)$ can be simply estimated by $q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \,|\, \mathbf{x}_o)$. In practice, a large number of simulations may be required for there to be enough training data in the vicinity of $\mathbf{x}_o$ in order to obtain an accurate posterior fit. However, running the simulator many times can be prohibitively expensive.

Sequential Neural Posterior Estimation is a strategy for reducing the number of simulations needed by conditional neural density estimation, originally proposed by Papamakarios and Murray [58] and further developed by Lueckmann et al. [42]. The name SNPE was first used for the method of Lueckmann et al. [42], but in this paper we will use it to refer to both methods, due to their close relationship. The main idea of SNPE is to generate parameter samples $\boldsymbol{\theta}_n$ from a proposal $\tilde{p}(\boldsymbol{\theta})$ instead of the prior $p(\boldsymbol{\theta})$ that makes generated data $\mathbf{x}_n$ more likely to be close to the observed datapoint $\mathbf{x}_o$. SNPE finds a good proposal $\tilde{p}(\boldsymbol{\theta})$ by training the estimator $q_{\boldsymbol{\phi}}$ over a number of rounds, whereby in each round $\tilde{p}(\boldsymbol{\theta})$ is taken to be the approximate posterior obtained in the round before. This sequential procedure converges rapidly and can be implemented with a relatively small number of simulations per round, which leads to massive savings in simulation cost. For its neural density estimator, SNPE uses a *Mixture Density Network* [7], which is a feedforward neural network that takes $\mathbf{x}$ as input and outputs the parameters of a Gaussian mixture over $\boldsymbol{\theta}$. To avoid overfitting due to the small number of training data per round, the MDN is trained with variational dropout [35].

The main issue with SNPE is that the proposal biases the approximation of the posterior. Since the parameter samples follow $\tilde{p}(\boldsymbol{\theta})$ instead of $p(\boldsymbol{\theta})$, the MDN will approximate $p(\mathbf{x}_o \,|\, \boldsymbol{\theta})\, \tilde{p}(\boldsymbol{\theta})$ instead of $p(\mathbf{x}_o \,|\, \boldsymbol{\theta})\, p(\boldsymbol{\theta})$ (up to normalization). Hence, an adjustment of either the learned posterior or the proposed samples must be made to account for sampling from the 'wrong' prior. In the variant of Papamakarios and Murray [58], which we refer to as SNPE-A, the learned posterior $q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \,|\, \mathbf{x}_o)$ is adjusted, by dividing it by $\tilde{p}(\boldsymbol{\theta})$ and multiplying it by $p(\boldsymbol{\theta})$. SNPE-A restricts $\tilde{p}(\boldsymbol{\theta})$ to be Gaussian; since $q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \,|\, \mathbf{x}_o)$ is a Gaussian mixture, the division by $\tilde{p}(\boldsymbol{\theta})$ can be done analytically. The problem with SNPE-A is that if $\tilde{p}(\boldsymbol{\theta})$ happens to have a smaller variance than any of the components of $q_{\boldsymbol{\phi}}(\boldsymbol{\theta} \,|\, \mathbf{x}_o)$, the division yields a Gaussian with negative variance, from which the algorithm is unable to recover and thus is forced to terminate prematurely. In the variant of Lueckmann et al. [42], which we refer to as SNPE-B, the parameter samples $\boldsymbol{\theta}_n$ are adjusted, by assigning them weights $w_n = p(\boldsymbol{\theta}_n)/\tilde{p}(\boldsymbol{\theta}_n)$. During training, the weighted log likelihood $\sum_n w_n \log q_{\boldsymbol{\phi}}(\boldsymbol{\theta}_n \,|\, \mathbf{x}_n)$ is used instead of the total log likelihood. Compared to SNPE-A, this method does not require the proposal $\tilde{p}(\boldsymbol{\theta})$ to be Gaussian, and it does not suffer from negative variances. However, the weights can have high variance, which may result in high-variance gradients and instability during training.

3 Sequential Neural Likelihood

Our new method, *Sequential Neural Likelihood (SNL)*, avoids the bias introduced by the proposal, by opting to learn a model of the likelihood instead of the posterior. Let $\tilde{p}(\boldsymbol{\theta})$ be a proposal distribution over parameters (not necessarily the prior) and let $p(\mathbf{x}\,|\,\boldsymbol{\theta})$ be the intractable likelihood of a simulator model. Consider a set of samples $\{\boldsymbol{\theta}_n, \mathbf{x}_n\}_{1:N}$ obtained by $\boldsymbol{\theta}_n \sim \tilde{p}(\boldsymbol{\theta})$ and $\mathbf{x}_n \sim p(\mathbf{x}\,|\,\boldsymbol{\theta}_n)$, and define $\tilde{p}(\boldsymbol{\theta}, \mathbf{x}) = p(\mathbf{x}\,|\,\boldsymbol{\theta})\,\tilde{p}(\boldsymbol{\theta})$ to be the joint distribution of each pair $(\boldsymbol{\theta}_n, \mathbf{x}_n)$. Suppose we train a conditional neural density estimator $q_{\boldsymbol{\phi}}(\mathbf{x}\,|\,\boldsymbol{\theta})$, which models the conditional of data given parameters, on the set $\{\boldsymbol{\theta}_n, \mathbf{x}_n\}_{1:N}$. For large N, maximizing the total log likelihood $\sum_n \log q_{\boldsymbol{\phi}}(\mathbf{x}_n\,|\,\boldsymbol{\theta}_n)$ is approximately equivalent to maximizing:

$$\begin{aligned}&\mathbb{E}_{\tilde{p}(\boldsymbol{\theta},\mathbf{x})}(\log q_{\boldsymbol{\phi}}(\mathbf{x}\,|\,\boldsymbol{\theta})) = \\ &\quad -\mathbb{E}_{\tilde{p}(\boldsymbol{\theta})}(D_{\mathrm{KL}}(p(\mathbf{x}\,|\,\boldsymbol{\theta})\,\|\,q_{\boldsymbol{\phi}}(\mathbf{x}\,|\,\boldsymbol{\theta}))) + \text{const}, \qquad (1)\end{aligned}$$

where $D_{\mathrm{KL}}(\cdot\,\|\,\cdot)$ is the Kullback–Leibler divergence. The above quantity attains its maximum when the KL is zero in the support of $\tilde{p}(\boldsymbol{\theta})$, i.e. when $q_{\boldsymbol{\phi}}(\mathbf{x}\,|\,\boldsymbol{\theta}) = p(\mathbf{x}\,|\,\boldsymbol{\theta})$ for all $\boldsymbol{\theta}$ such that $\tilde{p}(\boldsymbol{\theta}) > 0$. Therefore, given enough simulations, a sufficiently flexible conditional neural density estimator will eventually approximate the likelihood in the support of the proposal, regardless of the shape of the proposal. In other words, as long as we do not exclude parts of the parameter space, the way we propose parameters does not bias learning the likelihood asymptotically. Unlike when learning the posterior, no adjustment is necessary to account for our proposing strategy.

In practice, for a moderate number of simulations, the proposal $\tilde{p}(\boldsymbol{\theta})$ controls where $q_{\boldsymbol{\phi}}(\mathbf{x}\,|\,\boldsymbol{\theta})$ will be most accurate. In a parameter region where $\tilde{p}(\boldsymbol{\theta})$ is high, there will be a high concentration of training data, hence $p(\mathbf{x}\,|\,\boldsymbol{\theta})$ will be approximated better. Since we are ultimately interested in estimating the posterior $p(\boldsymbol{\theta}\,|\,\mathbf{x}_o)$ for a specific datapoint $\mathbf{x}_o$, it makes sense to use a proposal that is high in regions of high posterior density, and low otherwise, to avoid expending simulations in regions that are not relevant to the inference task. Inspired by SNPE, we train $q_{\boldsymbol{\phi}}$ over multiple rounds, indexed by $r \geq 1$. Let $\hat{p}_{r-1}(\boldsymbol{\theta}\,|\,\mathbf{x}_o)$ be the approximate posterior obtained in round $r-1$, and take $\hat{p}_0(\boldsymbol{\theta}\,|\,\mathbf{x}_o) = p(\boldsymbol{\theta})$. In round r, we generate a new batch of N parameters $\boldsymbol{\theta}_n$ from $\hat{p}_{r-1}(\boldsymbol{\theta}\,|\,\mathbf{x}_o)$, and data $\mathbf{x}_n$ from $p(\mathbf{x}\,|\,\boldsymbol{\theta}_n)$. We then (re-)train $q_{\boldsymbol{\phi}}$ on all data generated in rounds 1 up to r, and set $\hat{p}_r(\boldsymbol{\theta}\,|\,\mathbf{x}_o) \propto q_{\boldsymbol{\phi}}(\mathbf{x}_o\,|\,\boldsymbol{\theta})\,p(\boldsymbol{\theta})$. This method, which we call *Sequential Neural Likelihood*, is detailed in Algorithm 1.

In round r, SNL effectively uses rN parameter samples from $\tilde{p}_r(\boldsymbol{\theta}) = \frac{1}{r}\sum_{i=0}^{r-1}\hat{p}_i(\boldsymbol{\theta}\,|\,\mathbf{x}_o)$. As the amount of training data grows in each round, $q_{\boldsymbol{\phi}}(\mathbf{x}\,|\,\boldsymbol{\theta})$ becomes a more accurate model of the likelihood in the region of high proposal density, and hence the approximate posterior $\hat{p}_r(\boldsymbol{\theta}\,|\,\mathbf{x}_o)$ gets closer to the exact posterior. In turn, the proposal $\tilde{p}_r(\boldsymbol{\theta})$ also tends to the exact posterior, and therefore most of the simulations in later rounds come from parameter regions of high posterior density. In Section 5 we see that focusing on high posterior parameters massively reduces the number of simulations. Finally, unlike SNPE which trains only on simulations from the latest round, SNL trains on all simulations obtained up to each round.

Algorithm 1: Sequential Neural Likelihood (SNL)

```
Input  : observed data $\mathbf{x}_o$, estimator $q_{\boldsymbol{\phi}}(\mathbf{x}\,|\,\boldsymbol{\theta})$,
         number of rounds $R$, simulations per
         round $N$
Output : approximate posterior $\hat{p}(\boldsymbol{\theta}\,|\,\mathbf{x}_o)$
set $\hat{p}_0(\boldsymbol{\theta}\,|\,\mathbf{x}_o) = p(\boldsymbol{\theta})$ and $\mathcal{D} = \{\}$
for $r = 1 : R$ do
    for $n = 1 : N$ do
        sample $\boldsymbol{\theta}_n \sim \hat{p}_{r-1}(\boldsymbol{\theta}\,|\,\mathbf{x}_o)$ with MCMC
        simulate $\mathbf{x}_n \sim p(\mathbf{x}\,|\,\boldsymbol{\theta}_n)$
        add $(\boldsymbol{\theta}_n, \mathbf{x}_n)$ into $\mathcal{D}$
    (re-)train $q_{\boldsymbol{\phi}}(\mathbf{x}\,|\,\boldsymbol{\theta})$ on $\mathcal{D}$ and set
    $\hat{p}_r(\boldsymbol{\theta}\,|\,\mathbf{x}_o) \propto q_{\boldsymbol{\phi}}(\mathbf{x}_o\,|\,\boldsymbol{\theta})\,p(\boldsymbol{\theta})$
return $\hat{p}_R(\boldsymbol{\theta}\,|\,\mathbf{x}_o)$
```

In general, we are free to choose any neural density estimator $q_{\boldsymbol{\phi}}(\mathbf{x}\,|\,\boldsymbol{\theta})$ that is suitable for the task at hand. Because we are interested in a general-purpose solution, we propose taking $q_{\boldsymbol{\phi}}(\mathbf{x}\,|\,\boldsymbol{\theta})$ to be a conditional *Masked Autoregressive Flow* [59], which has been shown to perform well in a variety of general-purpose density estimation tasks. MAF represents $q_{\boldsymbol{\phi}}(\mathbf{x}\,|\,\boldsymbol{\theta})$ as a transformation of a standard Gaussian density $\mathcal{N}(\mathbf{0}, \mathbf{I})$ through a series of K autoregressive functions $f_1, \ldots, f_K$ each of which depends on $\boldsymbol{\theta}$, that is:

$$\mathbf{x} = \mathbf{z_K} \quad \text{where} \quad \begin{array}{l}\mathbf{z}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \\ \mathbf{z}_k = f_k(\mathbf{z}_{k-1}, \boldsymbol{\theta}).\end{array} \qquad (2)$$

Each f_k is a bijection with a lower-triangular Jacobian matrix, and is implemented by a *Masked Autoencoder for Distribution Estimation* [23] conditioned on $\boldsymbol{\theta}$. By change of variables, the conditional density is given by $q_{\boldsymbol{\phi}}(\mathbf{x}\,|\,\boldsymbol{\theta}) = \mathcal{N}(\mathbf{z}_0\,|\,\mathbf{0}, \mathbf{I}) \prod_k \left|\det\left(\frac{\partial f_k}{\partial \mathbf{z}_{k-1}}\right)\right|^{-1}$.

In order to sample from the approximate posterior $\hat{p}(\boldsymbol{\theta}\,|\,\mathbf{x}_o) \propto q_{\boldsymbol{\phi}}(\mathbf{x}_o\,|\,\boldsymbol{\theta})\,p(\boldsymbol{\theta})$, we use Markov Chain Monte Carlo (MCMC) in the form of Slice Sampling with axis-aligned updates [54]. The Markov chain is initialized with a sample from the prior and it persists across rounds; that is, in every round other than the first, the initial state of the chain is the same as its last state in the round before. At the beginning of each round, the chain is burned-in for 200 iterations, in order to adapt to the new approximate posterior. This scheme requires no tuning, and we found that it performed robustly across our experiments, so we recommend it as a default for general use, at least for tens of parameters that do not have pathologically strong correlations.

For parameter spaces of larger dimensionality, other MCMC schemes could be considered, such as Hamiltonian Monte Carlo [55].

4 Related work

Approximate Bayesian Computation [4, 5, 8, 41, 45]. ABC is a family of mainly non-parametric methods that repeatedly simulate the model, and reject simulations that do not reproduce the observed data. In practice, to reduce rejection rates to manageable levels, (a) lower-dimensional features (or summary statistics) are used instead of raw data, and (b) simulations are accepted whenever simulated features are within a distance ϵ from the observed features. A basic implementation of ABC would simulate parameters from the prior; more sophisticated variants such as *Markov Chain Monte Carlo ABC* [46, 50, 68] and *Sequential Monte Carlo ABC* [6, 10, 69, 74] guide future simulations based on previously accepted parameters. An issue with ABC in general is that in practice the required number of simulations increases dramatically as ϵ becomes small, which, as we shall see in Section 5, can lead to an unfavourable trade-off between estimation accuracy and simulation cost. Advanced ABC algorithms that work for $\epsilon = 0$ exist [26], but require the simulator to be differentiable.

Learning the posterior. Another approach to likelihood-free inference is learning a parametric model of the posterior from simulations. *Regression Adjustment* [5, 9] employs a parametric regressor (such as a linear model or a neural network) to learn the dependence of parameters $\boldsymbol{\theta}$ given data $\mathbf{x}$ in order to correct posterior parameter samples (obtained by e.g. ABC). *Gaussian Copula ABC* [39] estimates the posterior with a parametric Gaussian copula model. *Variational Likelihood-Free Inference* [51, 75, 76] trains a parametric posterior model by maximizing the (here intractable) variational lower bound; either the bound or its gradients are stochastically estimated from simulations. Parametrically learning the posterior is the target of *Sequential Neural Posterior Estimation* [42, 58], which SNL directly builds on, and which was discussed in detail in Section 2. Beyond SNPE, conditional neural density estimators have been used to learn the posterior from simulations in graphical models [52, 57], and universal probabilistic programs [37].

Learning the likelihood. Similarly to SNL, a body of work has focused on parametrically approximating the intractable likelihood function. *Synthetic Likelihood* [20, 56, 62, 83] estimates the mean $\mathbf{m}_{\boldsymbol{\theta}}$ and covariance matrix $\mathbf{S}_{\boldsymbol{\theta}}$ of a batch of data $\mathbf{x}_n \sim p(\mathbf{x} \,|\, \boldsymbol{\theta})$ sampled at a given $\boldsymbol{\theta}$, and then approximates $p(\mathbf{x}_o \,|\, \boldsymbol{\theta}) \approx \mathcal{N}(\mathbf{x}_o \,|\, \mathbf{m}_{\boldsymbol{\theta}}, \mathbf{S}_{\boldsymbol{\theta}})$. Non-Gaussian likelihood approximations are also possible, e.g. saddlepoint approximations [22]. Typically, SL would be run as an inner loop in the context of an MCMC sampler. *Gaussian Process Surrogate ABC* [48] employs a Gaussian process to model the dependence of $(\mathbf{m}_{\boldsymbol{\theta}}, \mathbf{S}_{\boldsymbol{\theta}})$ on $\boldsymbol{\theta}$, and uses the uncertainty in the GP to decide whether to run more simulations to estimate $(\mathbf{m}_{\boldsymbol{\theta}}, \mathbf{S}_{\boldsymbol{\theta}})$. Other predecessors to SNL include approximating the likelihood as a linear-Gaussian model [38], or as a mixture of Gaussian copulas with marginals modelled by mixtures of experts [21].

Learning the likelihood ratio. Rather than learning the likelihood $p(\mathbf{x} \,|\, \boldsymbol{\theta})$, an alternative approach is learning the likelihood ratio $p(\mathbf{x} \,|\, \boldsymbol{\theta}_1)/p(\mathbf{x} \,|\, \boldsymbol{\theta}_2)$ [11, 12, 15, 29, 60, 72], or the likelihood-to-marginal ratio $p(\mathbf{x} \,|\, \boldsymbol{\theta})/p(\mathbf{x})$ [19, 32, 75]. In practice, this can be done by training a classifier to discriminate between data simulated at $\boldsymbol{\theta}_1$ vs $\boldsymbol{\theta}_2$ for the likelihood ratio, or between data simulated at $\boldsymbol{\theta}$ vs random parameter samples from the prior for the likelihood-to-marginal ratio. The likelihood ratio can be used directly for Bayesian inference (e.g. by multiplying it with the prior and then sampling with MCMC), for hypothesis testing [11, 15], or for estimating the variational lower bound in the context of variational inference [75]. We note that the strategy for guiding simulations proposed in this paper can (at least in principle) be used with likelihood-ratio estimation too; it might be possible that large computational savings can be achieved as in SNL.

Guiding simulations. Various approaches have been proposed for guiding simulations in order to reduce simulation cost, typically by making use of the observed data $\mathbf{x}_o$ in some way. *Optimization Monte Carlo* [49] directly optimizes parameter samples so as to generate data similar to $\mathbf{x}_o$. *Bayesian Optimization Likelihood-free Inference* [28] uses Bayesian optimization to find parameter samples that minimize the distance $\|\mathbf{x} - \mathbf{x}_o\|$. An active-learning-style approach is to use the Bayesian uncertainty in the posterior estimate to choose what simulation to run next [33, 43]; in this scheme, the next simulation is chosen to maximally reduce the Bayesian uncertainty. Similarly to *Sequential Neural Posterior Estimation* [42, 58], SNL selects future simulations by proposing parameters from preliminary approximations to the posterior, which in Section 5 is shown to reduce simulation cost significantly.

5 Experiments

5.1 Setup

In all experiments, we used a Masked Autoregressive Flow (MAF) with 5 autoregressive layers, each of which has two hidden layers of 50 units each and tanh nonlinearities. Our design and training choices follow closely the reference implementation [59]. We used batch normalization [31] between autoregressive layers, and trained MAF by stochastically maximizing the total log likelihood using Adam [34] with a minibatch size of 100 and a learning rate of 10^{-4}. We used 1000 simulations in each round of SNL, 5% of which were randomly selected to be used as a validation set; we stopped training if validation log likelihood did not improve after 20 epochs. No other form of regularization was used other than early stopping. These settings were held constant and performed robustly across all experiments, which is evidence for the robustness of SNL to parameter tuning. We recommend these settings as defaults for general use.

We compare SNL to the following algorithms:

Neural Likelihood (NL). By this we refer to training a MAF (of the same architecture as above) on N simulations from the prior. This is essentially SNL without simulation guiding; we use it as a control to assess the benefit of SNL's guiding strategy. We vary N from 10^3 to 10^6 (or more if the simulation cost permits it), and plot the performance for each N.

SNPE-A [58]. We use 1000 simulations in each of the proposal-estimating rounds, and 2000 simulations in the posterior-estimating round (the final round). In all experiments, SNPE-A uses a Mixture Density Network [7] with two hidden layers of 50 units each, 8 mixture components, and tanh nonlinearities. We chose the MDN to have roughly as many parameters as MAF in SNL. The MDN is trained for 1000 epochs per round, except for the final round which uses 5000 epochs, and is regularized with variational dropout [35].

SNPE-B [42]. We use 1000 simulations in each round. The same MDN as in SNPE-A is used, and it is trained for 1000 epochs per round using variational dropout.

Synthetic Likelihood (SL) [83]. Our implementation uses axis-aligned Slice Sampling [54], where the intractable likelihood is approximated by a Gaussian fitted to a batch of N simulations run on the fly at each visited parameter $\boldsymbol{\theta}$. We vary N from 10 to 100–1000 (depending on what the simulation cost of each experiment permits) and plot the performance of SL for each N. We run Slice Sampling until we obtain 1000 posterior samples.

Sequential Monte Carlo ABC (SMC-ABC). We use the version of Beaumont et al. [6]. We use 1000 particles, and resample the population if the effective sample size falls below 50%. We set the initial ϵ such that the acceptance probability in the first round is at least 20%, and decay ϵ by a factor of 0.9 per round.

We chose SNPE-A/B because they are the most related methods, and because they achieve state-of-the-art results in the literature; the comparison with them is intended to establish the competitiveness of SNL. The versions of SL and SMC-ABC we use here are not state-of-the-art, but they are robust and widely-used, and are intended as baselines. The comparison with them is meant to demonstrate the gap between state-of-the-art neural-based methods and off-the-shelf, commonly-used alternatives.

5.2 Results

We demonstrate SNL in four cases: (a) a toy model with complex posterior, (b) an M/G/1 queue model, (c) a Lotka–Volterra model from ecology, and (d) a Hodgkin–Huxley model of neural activity from neuroscience. The first two models are fast to simulate, and are intended as toy demonstrations. The last two models are relatively slow to simulate (as they involve numerically solving differential equations), and are illustrative of real-world problems of interest. In the last two models, the computational cost of training the neural networks for SNPE-A/B and SNL is negligible compared to the cost of simulating training data.

A detailed description of the models and the full set of results are in Appendices A and B. Code that reproduces the experiments with detailed user instructions can be found at `https://github.com/gpapamak/snl`.

A toy model with complex posterior. We consider the following model, where $\boldsymbol{\theta}$ is 5-dimensional, and $\mathbf{x}$ is a set of four 2-dimensional points (or an 8-dimensional vector) sampled from a Gaussian whose mean $\mathbf{m}_{\boldsymbol{\theta}}$ and covariance matrix $\mathbf{S}_{\boldsymbol{\theta}}$ are functions of $\boldsymbol{\theta}$:

$$\theta_i \sim \mathcal{U}(-3,3) \quad \text{for} \quad i = 1,\ldots,5 \tag{3}$$

$$\mathbf{m}_{\boldsymbol{\theta}} = (\theta_1, \theta_2) \tag{4}$$

$$s_1 = \theta_3^2, \quad s_2 = \theta_4^2, \quad \rho = \tanh(\theta_5) \tag{5}$$

$$\mathbf{S}_{\boldsymbol{\theta}} = \begin{pmatrix} s_1^2 & \rho s_1 s_2 \\ \rho s_1 s_2 & s_2^2 \end{pmatrix} \tag{6}$$

$$\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_4) \quad \text{where} \quad \mathbf{x}_j \sim \mathcal{N}(\mathbf{m}_{\boldsymbol{\theta}}, \mathbf{S}_{\boldsymbol{\theta}}). \tag{7}$$

The likelihood is $p(\mathbf{x}\,|\,\boldsymbol{\theta}) = \prod_j \mathcal{N}(\mathbf{x}_j\,|\,\mathbf{m}_{\boldsymbol{\theta}}, \mathbf{S}_{\boldsymbol{\theta}})$. Despite the model's simplicity, the posterior is complex and non-trivial: it has four symmetric modes (due to squaring), and vertical cut-offs (due to the uniform prior). This example illustrates that the posterior can be complicated even if the prior and the likelihood are composed entirely of simple operations (which is a common situation e.g. in probabilistic programming). In such situations, approximating the likelihood can be simpler than approximating the posterior.

Figure 1a shows the Maximum Mean Discrepancy (MMD) [27] between the approximate posterior of each method and the true posterior vs the total number of simulations used. SNL achieves the best trade-off

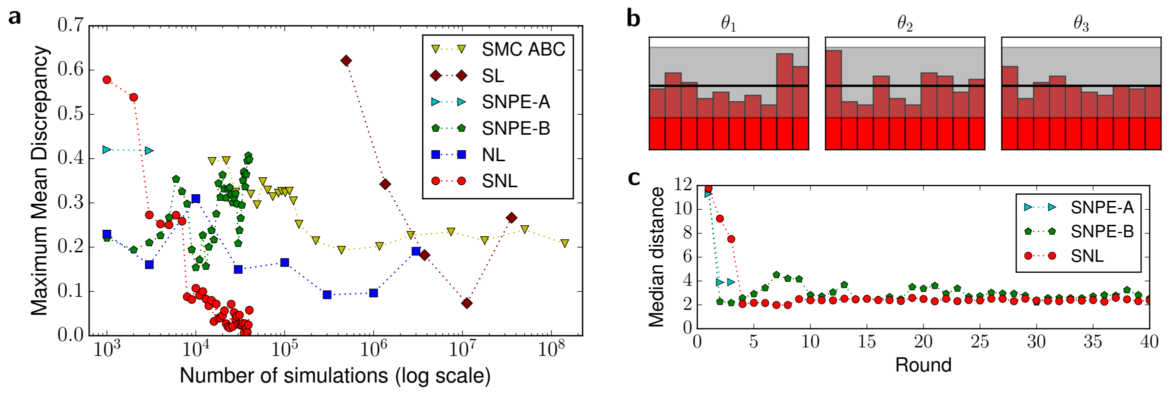

Figure 1: A toy model with complex posterior. **a**: Accuracy vs simulation cost: bottom left is best. **b**: Calibration test for SNL, histogram outside gray band indicates poor calibration. **c**: Median distance from simulated to observed data.

between accuracy and simulation cost, and achieves the most accurate approximation to the posterior overall. SNPE-A fails in the second round due to the variance of the proposal becoming negative (hence there are only two points in the graph) and SNPE-B experiences high variability; SNL is significantly more robust in comparison. SMC-ABC and SL require orders of magnitude more simulations than the sequential neural methods.

To assess whether SNL is well-calibrated, we performed a simulation-based calibration test [73]: we generated 200 pairs $(\boldsymbol{\theta}_n, \mathbf{x}_n)$ from the joint $p(\boldsymbol{\theta}, \mathbf{x})$, used SNL to approximate the posterior for each $\mathbf{x}_n$, obtained 9 close-to-independent samples from each posterior, and calculated the rank statistic of each parameter θ_{ni} for $i = 1, \ldots, 5$ in the corresponding set of posterior samples. If SNL is well-calibrated, the distribution of each rank statistic must be uniform. The histograms of the first three rank statistics are shown in Figure 1b, with a gray band showing the expected variability of a uniform histogram. The test does not find evidence of any gross mis-calibration, and suggests that SNL performs consistently across multiple runs (here 200).

Another diagnostic is shown in Figure 1c, where we plot the median distance between simulated and observed data for each round. From this plot we can assess convergence, and determine the minimum number of rounds to run for. SNL has lower median distance compared to SNPE-B, which is evidence that SNPE-B has not estimated the posterior accurately enough (as also shown in the left plot).

M/G/1 queue model [67]. The model describes a server processing customers waiting in a queue, and has parameters $\boldsymbol{\theta} = (\theta_1, \theta_2, \theta_3)$. The time it takes to serve a customer is uniformly distributed in $[\theta_1, \theta_2]$, and the time between customer arrivals is exponentially distributed with rate θ_3. The data $\mathbf{x}$ is 5 equally spaced quantiles of the distribution of inter-departure times. The model can be easily simulated, but the likelihood $p(\mathbf{x} \,|\, \boldsymbol{\theta})$ is intractable. Our experimental setup follows Papamakarios and Murray [58].

The trade-off between accuracy and simulation cost is shown in Figure 2a. Here we do not have access to the true posterior; instead we plot the negative log probability of the true parameters, obtained by kernel density estimation on posterior samples, vs number of simulations. SNL recovers the true parameters faster and more accurately than all other methods. Although high posterior probability of true parameters is not enough evidence that the posterior is correctly calibrated (e.g. it could be that the approximate posterior happens to be centred at the right value, but is overconfident), the calibration test (Figure 2b) suggests that SNL is indeed well-calibrated.

Another diagnostic possible with SNL is to check how well $q_{\boldsymbol{\phi}}(\mathbf{x} \,|\, \boldsymbol{\theta})$ fits the data distribution $p(\mathbf{x} \,|\, \boldsymbol{\theta})$ for a certain value of $\boldsymbol{\theta}$. To do this we (a) generate N independent samples from $p(\mathbf{x} \,|\, \boldsymbol{\theta})$ by running the simulator, (b) generate N independent samples from $q_{\boldsymbol{\phi}}(\mathbf{x} \,|\, \boldsymbol{\theta})$ using Equation (2), and (c) calculate the Maximum Mean Discrepancy [27] between the two sets of samples. This is shown in Figure 2c using the true parameters, where we compare with NL and a baseline Gaussian directly fitted to the samples from $p(\mathbf{x} \,|\, \boldsymbol{\theta})$. This plot can be used to assess the performance of SNL, and also determine how many rounds to run SNL for. We note that this kind of diagnostic is not possible with methods that approximate the posterior or the likelihood ratio instead of the likelihood.

Lotka–Volterra population model [82]. This is a Markov jump process that models the interaction of a population of predators with a population of prey. It is a classic model of oscillating populations. It has four

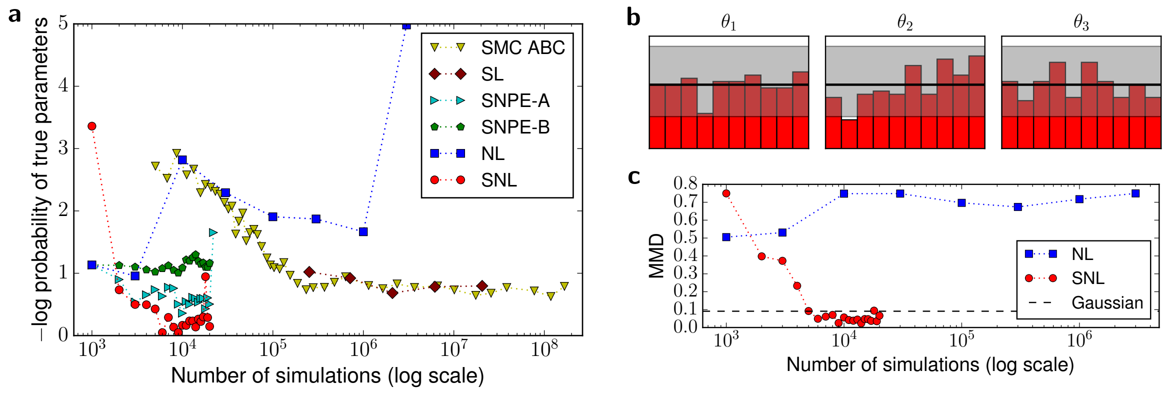

Figure 2: M/G/1 queue model. **a**: Accuracy vs simulation cost: bottom left is best. **b**: Calibration test for SNL, histogram outside gray band indicates poor calibration. **c**: Likelihood goodness-of-fit vs simulation cost, calculated at true parameters.

parameters $\boldsymbol{\theta}$, which control the rate of (a) predator births, (b) predator deaths, (c) prey births, and (d) predator-prey interactions. The process can be simulated exactly (by using e.g. the Gillespie algorithm [24]), but its likelihood is intractable. In our experiments we follow the setup of Papamakarios and Murray [58], where $\mathbf{x}$ is 9 features of the population timeseries.

Figure 3a shows negative log probability of true parameters vs simulation cost. SNL and SNPE-A perform the best, whereas SNPE-B is less accurate. Running the calibration test for SNL, using parameters drawn from a broad prior, shows the procedure is sometimes over-confident (Figure 3b, left plot). In this test, many of the 'true' parameters considered corresponded to uninteresting models, where both populations died out quickly, or the prey population diverged. In an application we want to know if the procedure is well-behaved for the sorts of parameters we seem to have. We ran a calibration test for an alternative prior, constrained to parameters that give the oscillating behaviour observed in interesting data. This test suggests that the calibration is reasonable when modelling oscillating data (Figure 3b, right plot). If we had still observed calibration problems we would have investigated using larger neural networks and/or longer MCMC runs in SNL. Figure 3c shows the median distance between the simulated data and the observed data for each round of SNPE-A, SNPE-B and SNL. We see that SNPE-A and SNL have a lower median distance compared to SNPE-B, which suggests that SNPE-B has not estimated the posterior accurately enough.

Hodgkin–Huxley neuron model [30]. This model describes how the electrical potential measured across the cell membrane of a neuron varies over time as a function of current injected through an electrode. We used a model of a regular-spiking cortical pyramidal cell [61]; the model is described by a set of five coupled ordinary differential equations, which we solved numerically using NEURON [13]. This is a challenging problem of practical interest in neuroscience [17, 42], whose task is to infer 12 parameters describing the function of the neuron from features of the membrane potential timeseries. We followed the setup of Lueckmann et al. [42], and we used 18 features extracted from the timeseries as data $\mathbf{x}$.

Figure 4a shows negative log probability of true parameters vs simulation cost; SNL outperforms all other methods. SNPE-A fails in the second round due to the variance of the proposal becoming negative. The calibration test in Figure 4b suggests that SNL is well-calibrated. Figure 4c shows goodness-of-fit between MAF and the true likelihood as measured by MMD; SNL converges faster than NL to the true likelihood. The comparison with a baseline Gaussian fit is evidence that SNL has not fully converged, which indicates that running SNL for longer may improve results further.

6 Discussion

Performance and robustness of SNL. Our experiments show that SNL is accurate, efficient and robust. The comparison between SNL and NL demonstrates that guiding the simulations is crucial in achieving a good likelihood fit in the region of interest with a reasonable number of simulations. The comparison between SNL and SMC-ABC shows that flexible parametric models can be more accurate and cost-effective compared to non-parametric alternatives. The comparison with SNPE shows that SNL is more robust; SNPE-A failed in two out of four cases, and SNPE-B exhibited high variability. We used the same MAF architecture and training hyperparameters in all our experiments, which shows that a flexible neural architecture can be broadly applicable without extensive

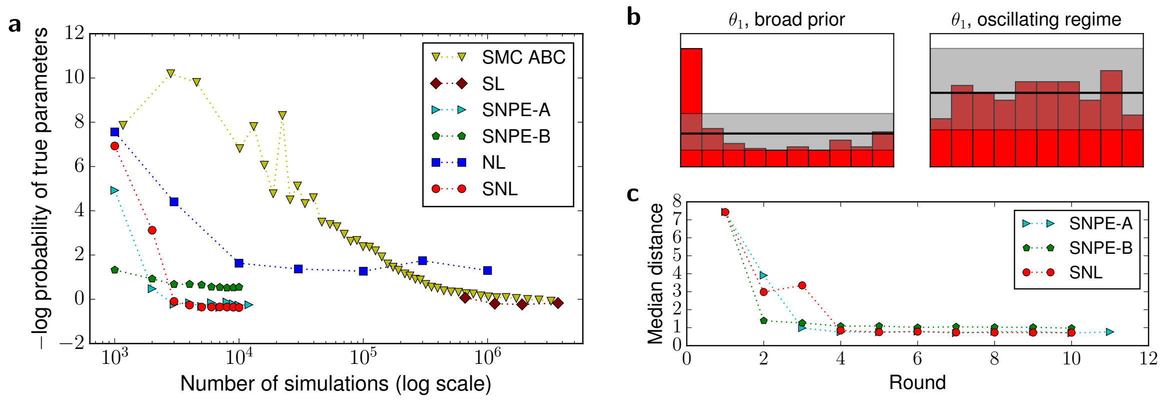

Figure 3: Lotka–Volterra model. **a**: Accuracy vs simulation cost: bottom left is best. **b**: Calibration tests for SNL, histogram outside gray band indicates poor calibration. Calibration depends on data regime (see main text). **c**: Median distance from simulated to observed data.

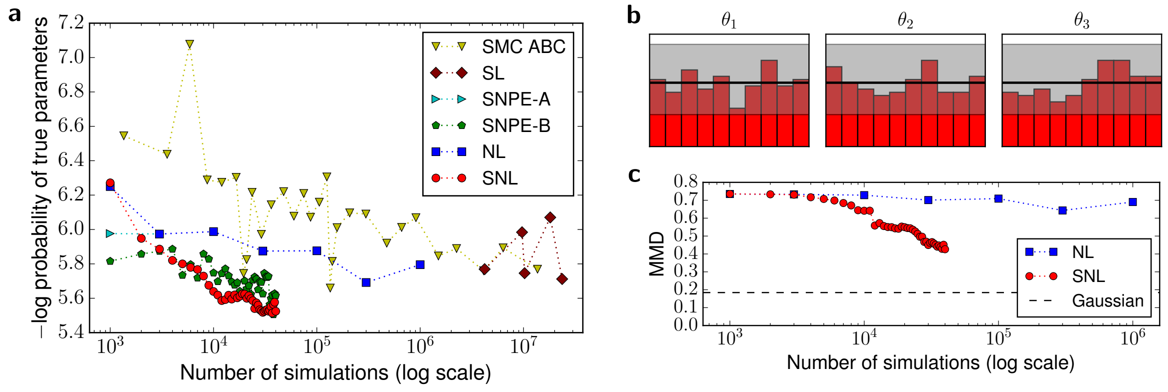

Figure 4: Hodgkin–Huxley model. **a**: Accuracy vs simulation cost: bottom left is best. **b**: Calibration test for SNL, histogram outside gray band indicates poor calibration. **c**: Likelihood goodness-of-fit vs simulation cost, calculated at true parameters.

problem-specific tuning. We also discussed a number of diagnostics that can be used with SNL to determine the number of rounds to run it for, assess calibration, diagnose convergence and check goodness-of-fit.

Scaling to high-dimensional data. Likelihood-free inference becomes challenging when the dimensionality of the data $\mathbf{x}$ is large, which in practice necessitates the use of low-dimensional features (or summary statistics) in place of the raw data. SNL relies on estimating the density of the data, which is a hard problem in high dimensions. A potential strategy for scaling SNL up to high dimensions is exploiting the structure of the data. For instance, if $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$ is a dataset of a large number of i.i.d. datapoints, we could decompose the likelihood as $p(\mathbf{x} \mid \boldsymbol{\theta}) = \prod_n p(\mathbf{x}_n \mid \boldsymbol{\theta})$ and only learn a model $q_{\boldsymbol{\phi}}(\mathbf{x}_n \mid \boldsymbol{\theta})$ of the density in the lower-dimensional space of individual points. By exploiting data structure, neural density estimators have been shown to accurately model high-dimensional data such as images [18, 78, 79] and raw audio [77, 80]; SNL could easily incorporate such or further advances in neural density estimation for high-dimensional structured objects.

Learning the likelihood vs the posterior. A general question is whether it is preferable to learn the posterior or the likelihood; SNPE learns the posterior, whereas SNL targets the likelihood. As we saw, learning the likelihood can often be easier than learning the posterior, and it does not depend on the choice of proposal, which makes learning easier and more robust. Moreover, a model of the likelihood can be reused with different priors, and is in itself an object of interest that can be used for identifiability analysis [40] or hypothesis testing [11, 15]. On the other hand, methods such as SNPE return a parametric model of the posterior directly, whereas a further inference step (e.g. variational inference or MCMC) is needed on top of SNL to obtain a posterior estimate, which introduces further computational cost and approximation error. Ultimately, the best approach depends on the problem and application at hand.

Acknowledgments

We thank Kyle Cranmer and Michael Gutmann for useful discussions, and James Ritchie for pointing us to simulation-based calibration [73]. George Papamakarios was supported by the Centre for Doctoral Training in Data Science, funded by EPSRC (grant EP/L016427/1) and the University of Edinburgh, and by Microsoft Research through its PhD Scholarship Programme. David C. Sterratt received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 785907.

References

[1] S. Agostinelli et al. Geant4—a simulation toolkit. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 506(3):250–303, 2003.

[2] J. Alsing, B. Wandelt, and S. Feeney. Massive optimal data compression and density estimation for scalable, likelihood-free inference in cosmology. *Monthly Notices of the Royal Astronomical Society*, 477(3):2874–2885, 2018.

[3] R. Bansal and A. Yaron. Risks for the long run: A potential resolution of asset pricing puzzles. *Journal of Finance*, 59(4):1481–1509, 2004.

[4] M. A. Beaumont. Approximate Bayesian computation in evolution and ecology. *Annual Review of Ecology, Evolution, and Systematics*, 41(1):379–406, 2010.

[5] M. A. Beaumont, W. Zhang, and D. J. Balding. Approximate Bayesian computation in population genetics. *Genetics*, 162:2025–2035, 2002.

[6] M. A. Beaumont, J.-M. Cornuet, J.-M. Marin, and C. P. Robert. Adaptive approximate Bayesian computation. *Biometrika*, 96(4):983–990, 2009.

[7] C. M. Bishop. Mixture density networks. Technical Report NCRG/94/004, Aston University, 1994.

[8] M. G. B. Blum. Approximate Bayesian computation: A nonparametric perspective. *Journal of the American Statistical Association*, 105(491): 1178–1187, 2010.

[9] M. G. B. Blum and O. François. Non-linear regression models for approximate Bayesian computation. *Statistics and Computing*, 20(1):63–73, 2010.

[10] F. V. Bonassi and M. West. Sequential Monte Carlo with adaptive weights for approximate Bayesian computation. *Bayesian Analysis*, 10(1): 171–187, 2015.

[11] J. Brehmer, K. Cranmer, G. Louppe, and J. Pavez. A guide to constraining effective field theories with machine learning. *Physical Review Letters*, 98(5): 052004, 2018.

[12] J. Brehmer, G. Louppe, J. Pavez, and K. Cranmer. Mining gold from implicit models to improve likelihood-free inference. *arXiv:1805.12244*, 2018.

[13] T. Carnevale and M. Hines. *The NEURON Book*. Cambridge University Press, 2006.

[14] M. L. Casado, A. G. Baydin, D. M. Rubio, T. A. Le, F. Wood, L. Heinrich, G. Louppe, K. Cranmer, K. Ng, W. Bhimji, and Prabhat. Improvements to inference compilation for probabilistic programming in large-scale scientific simulators. *arXiv:1712.07901*, 2017.

[15] K. Cranmer, J. Pavez, and G. Louppe. Approximating likelihood ratios with calibrated discriminative classifiers. *arXiv:1506.02169*, 2016.

[16] M. F. Cusumano-Towner, A. Radul, D. Wingate, and V. K. Mansinghka. Probabilistic programs for inferring the goals of autonomous agents. *arXiv:1704.04977*, 2017.

[17] A. C. Daly, D. J. Gavaghan, C. Holmes, and J. Cooper. Hodgkin–Huxley revisited: reparametrization and identifiability analysis of the classic action potential model with approximate Bayesian methods. *Royal Society Open Science*, 2(12), 2015.

[18] L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using Real NVP. *Proceedings of the 5th International Conference on Learning Representations*, 2017.

[19] R. Dutta, J. Corander, S. Kaski, and M. U. Gutmann. Likelihood-free inference by ratio estimation. *arXiv:1611.10242*, 2016.

[20] R. G. Everitt. Bootstrapped synthetic likelihood. *arXiv:1711.05825*, 2018.

[21] Y. Fan, D. J. Nott, and S. A. Sisson. Approximate Bayesian Computation via regression density estimation. *Stat*, 2(1):34–48, 2013.

[22] M. Fasiolo, S. N. Wood, F. Hartig, and M. V. Bravington. An extended empirical saddlepoint approximation for intractable likelihoods. *Electronic Journal of Statistics*, 12(1):1544–1578, 2018.

[23] M. Germain, K. Gregor, I. Murray, and H. Larochelle. MADE: Masked autoencoder for distribution estimation. *Proceedings of the 32nd International Conference on Machine Learning*, 2015.

[24] D. T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *The Journal of Physical Chemistry*, 81(25):2340–2361, 1977.

[25] C. Gouriéroux, A. Monfort, and E. Renault. Indirect inference. *Journal of Applied Econometrics*, 8(S1):S85–S118, 1993.

[26] M. M. Graham and A. J. Storkey. Asymptotically exact inference in differentiable generative models. *Electronic Journal of Statistics*, 11(2):5105–5164, 2017.

[27] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13 (1):723–773, 2012.

[28] M. U. Gutmann and J. Corander. Bayesian optimization for likelihood-free inference of simulator-based statistical models. *Journal of Machine Learning Research*, 17(125):1–47, 2016.

[29] M. U. Gutmann, R. Dutta, S. Kaski, and J. Corander. Likelihood-free inference via classification. *Statistics and Computing*, 28(2):411–425, 2018.

[30] A. L. Hodgkin and A. F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *The Journal of Physiology*, 117:500–544, 1952.

[31] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *Proceedings of the 32nd International Conference on Machine Learning*, 2015.

[32] R. Izbicki, A. Lee, and C. Schafer. High-dimensional density ratio estimation with extensions to approximate likelihood computation. *Proceedings of the 17th International Conference on Artificial Intelligence and Statistics*, 2014.

[33] M. Järvenpää, M. U. Gutmann, A. Pleska, A. Vehtari, and P. Marttinen. Efficient acquisition rules for model-based approximate Bayesian computation. *Bayesian Analysis*, 2018.

[34] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *Proceedings of the 3rd International Conference on Learning Representations*, 2015.

[35] D. P. Kingma, T. Salimans, and M. Welling. Variational dropout and the local reparameterization trick. *Advances in Neural Information Processing Systems 28*, 2015.

[36] T. D. Kulkarni, P. Kohli, J. B. Tenenbaum, and V. K. Mansinghka. Picture: A probabilistic programming language for scene perception. *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[37] T. A. Le, A. G. Baydin, and F. Wood. Inference compilation and universal probabilistic programming. *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, pages 1338–1348, 2017.

[38] C. Leuenberger and D. Wegmann. Bayesian computation and model selection without likelihoods. *Genetics*, 184(1):243–252, 2010.

[39] J. Li, D. J. Nott, Y. Fan, and S. A. Sisson. Extending approximate Bayesian computation methods to high dimensions via a Gaussian copula model. *Computational Statistics & Data Analysis*, 106(C): 77–89, 2017.

[40] J. Lintusaari, M. U. Gutmann, S. Kaski, and J. Corander. On the identifiability of transmission dynamic models for infectious diseases. *Genetics*, 202(3):911–918, 2016.

[41] J. Lintusaari, M. U. Gutmann, R. Dutta, S. Kaski, and J. Corander. Fundamentals and recent developments in approximate Bayesian computation. *Systematic Biology*, 66(1):e66–e82, 2017.

[42] J.-M. Lueckmann, P. J. Goncalves, G. Bassetto, K. Öcal, M. Nonnenmacher, and J. H. Macke. Flexible statistical inference for mechanistic models of neural dynamics. *Advances in Neural Information Processing Systems 30*, 2017.

[43] J.-M. Lueckmann, G. Bassetto, T. Karaletsos, and J. H. Macke. Likelihood-free inference with emulator networks. *arXiv:1805.09294*, 2018.

[44] V. K. Mansinghka, T. D. Kulkarni, Y. N. Perov, and J. B. Tenenbaum. Approximate Bayesian image interpretation using generative probabilistic graphics programs. *Advances in Neural Information Processing Systems 26*, 2013.

[45] J.-M. Marin, P. Pudlo, C. P. Robert, and R. J. Ryder. Approximate Bayesian computational methods. *Statistics and Computing*, 22(6):1167–1180, 2012.

[46] P. Marjoram, J. Molitor, V. Plagnol, and S. Tavaré. Markov chain Monte Carlo without likelihoods. *Proceedings of the National Academy of Sciences*, 100(26):15324–15328, 2003.

[47] H. Markram et al. Reconstruction and simulation of neocortical microcircuitry. *Cell*, 163:456–492, 2015.

[48] E. Meeds and M. Welling. GPS-ABC: Gaussian process surrogate approximate Bayesian computation. *Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence*, 2014.

[49] E. Meeds and M. Welling. Optimization Monte Carlo: Efficient and embarrassingly parallel likelihood-free inference. *Advances in Neural Information Processing Systems 28*, 2015.

[50] E. Meeds, R. Leenders, and M. Welling. Hamiltonian ABC. *Proceedings of the 31st Conference on Uncertainty in Artificial Intelligence*, 2015.

[51] A. Moreno, T. Adel, E. Meeds, J. M. Rehg, and M. Welling. Automatic variational ABC. *arXiv:1606.08549*, 2016.

[52] Q. Morris. Recognition networks for approximate inference in BN20 networks. *Proceedings of the 17th Conference on Uncertainty in Artificial Intelligence*, 2001.

[53] R. M. Neal. Probabilistic inference using Markov chain Monte Carlo methods. Technical Report CRG-TR-93-1, Department of Computer Science, University of Toronto, 1993.

[54] R. M. Neal. Slice sampling. *The Annals of Statistics*, 31(3):705–767, 2003.

[55] R. M. Neal. MCMC using Hamiltonian dynamics. In *Handbook of Markov Chain Monte Carlo*, chapter 5, pages 113–162. Chapman & Hall/CRC., 2011.

[56] V. M. H. Ong, D. J. Nott, M.-N. Tran, S. A. Sisson, and C. C. Drovandi. Variational Bayes with synthetic likelihood. *Statistics and Computing*, 28 (4):971–988, 2018.

[57] B. Paige and F. Wood. Inference networks for sequential Monte Carlo in graphical models. *Proceedings of the 33rd International Conference on Machine Learning*, 2016.

[58] G. Papamakarios and I. Murray. Fast ϵ-free inference of simulation models with Bayesian conditional density estimation. *Advances in Neural Information Processing Systems 29*, 2016.

[59] G. Papamakarios, T. Pavlakou, and I. Murray. Masked autoregressive flow for density estimation. *Advances in Neural Information Processing Systems 30*, 2017.

[60] K. C. Pham, D. J. Nott, and S. Chaudhuri. A note on approximating ABC-MCMC using flexible classifiers. *Stat*, 3(1):218–227, 2014.

[61] M. Pospischil, M. Toledo-Rodriguez, C. Monier, Z. Piwkowska, T. Bal, Y. Frégnac, H. Markram, and A. Destexhe. Minimal Hodgkin–Huxley type models for different classes of cortical and thalamic neurons. *Biological Cybernetics*, 99(4):427–441, 2008.

[62] L. F. Price, C. C. Drovandi, A. Lee, and D. J. Nott. Bayesian synthetic likelihood. *Journal of Computational and Graphical Statistics*, 27(1):1–11, 2018.

[63] J. K. Pritchard, M. T. Seielstad, A. Perez-Lezaun, and M. W. Feldman. Population growth of human Y chromosomes: a study of Y chromosome microsatellites. *Molecular Biology and Evolution*, 16(12):1791–1798, 1999.

[64] O. Ratmann, O. Jørgensen, T. Hinkley, M. Stumpf, S. Richardson, and C. Wiuf. Using likelihood-free inference to compare evolutionary dynamics of the protein networks of H. pylori and P. falciparum. *PLoS Computational Biology*, 3(11):2266–2278, 2007.

[65] L. Romaszko, C. K. Williams, P. Moreno, and P. Kohli. Vision-as-inverse-graphics: Obtaining a rich 3D explanation of a scene from a single image. *IEEE International Conference on Computer Vision Workshop*, 2017.

[66] C. M. Schafer and P. E. Freeman. Likelihood-free inference in cosmology: Potential for the estimation of luminosity functions. *Statistical Challenges in Modern Astronomy V*, pages 3–19, 2012.

[67] A. Y. Shestopaloff and R. M. Neal. On Bayesian inference for the M/G/1 queue with efficient MCMC sampling. *arXiv:1401.5548*, 2014.

[68] S. A. Sisson and Y. Fan. Likelihood-free Markov chain Monte Carlo. In *Handbook of Markov Chain Monte Carlo*, chapter 12, pages 313–336. Chapman & Hall/CRC., 2011.

[69] S. A. Sisson, Y. Fan, and M. M. Tanaka. Sequential Monte Carlo without likelihoods. *Proceedings of the National Academy of Sciences*, 104(6):1760–1765, 2007.

[70] T. Sjöstrand, S. Mrenna, and P. Skands. A brief introduction to PYTHIA 8.1. *Computer Physics Communications*, 178(11):852–867, 2008.

[71] D. C. Sterratt, B. Graham, A. Gillies, and D. Willshaw. *Principles of Computational Modelling in Neuroscience*. Cambridge University Press, 2011.

[72] M. Stoye, J. Brehmer, G. Louppe, J. Pavez, and K. Cranmer. Likelihood-free inference with an improved cross-entropy estimator. *arXiv:1808.00973*, 2018.

[73] S. Talts, M. Betancourt, D. Simpson, A. Vehtari, and A. Gelman. Validating Bayesian inference algorithms with simulation-based calibration. *arXiv:1804.06788*, 2018.

[74] T. Toni, D. Welch, N. Strelkowa, A. Ipsen, and M. P. H. Stumpf. Approximate Bayesian computation scheme for parameter inference and model selection in dynamical systems. *Journal of The Royal Society Interface*, 6(31):187–202, 2009.

[75] D. Tran, R. Ranganath, and D. Blei. Hierarchical implicit models and likelihood-free variational inference. *Advances in Neural Information Processing Systems 30*, 2017.

[76] M.-N. Tran, D. J. Nott, and R. Kohn. Variational Bayes with intractable likelihood. *Journal of Computational and Graphical Statistics*, 26(4):873–882, 2017.

[77] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu. WaveNet: A generative model for raw audio. *arXiv:1609.03499*, 2016.

[78] A. van den Oord, N. Kalchbrenner, L. Espeholt, K. Kavukcuoglu, O. Vinyals, and A. Graves. Conditional image generation with PixelCNN decoders. *Advances in Neural Information Processing Systems 29*, 2016.

[79] A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. *Proceedings of the 33rd International Conference on Machine Learning*, 2016.

[80] A. van den Oord et al. Parallel WaveNet: Fast high-fidelity speech synthesis. *Proceedings of the 35th International Conference on Machine Learning*, 2018.

[81] M. J. Wainwright and M. I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1–2):1–305, 2008.

[82] D. J. Wilkinson. *Stochastic Modelling for Systems Biology, Second Edition*. Taylor & Francis, 2011.

[83] S. N. Wood. Statistical inference for noisy nonlinear ecological dynamic systems. *Nature*, 466 (7310):1102–1104, 2010.

[84] J. Wu, I. Yildirim, J. J. Lim, B. Freeman, and J. B. Tenenbaum. Galileo: Perceiving physical object properties by integrating a physics engine with deep learning. *Advances in Neural Information Processing Systems 28*, 2015.

[85] A. Yuille and D. Kersten. Vision as Bayesian inference: analysis by synthesis? *Trends in Cognitive Sciences*, 10(7):301–308, 2006.