Simulation-based Inference for High-dimensional Data using Surjective Sequential Neural Likelihood Estimation

Simon Dirmeier[1,2] **Carlo Albert**[3] **Fernando Perez-Cruz**[2,4]

[1]Swiss Data Science Center, Switzerland
[2]ETH Zurich, Switzerland
[3]Swiss Federal Institute of Aquatic Science and Technology, Switzerland
[4]Bank for International Settlements, Switzerland

Abstract

Neural likelihood estimation methods for simulation-based inference can suffer from performance degradation when the modeled data is very high-dimensional or lies along a lower-dimensional manifold, which is due to the inability of the density estimator to accurately estimate a density function. We present Surjective Sequential Neural Likelihood (SSNL) estimation, a novel member in the family of methods for simulation-based inference (SBI). SSNL fits a dimensionality-reducing surjective normalizing flow model and uses it as a surrogate likelihood function, which allows for computational inference via Markov chain Monte Carlo or variational Bayes methods. Among other benefits, SSNL avoids the requirement to manually craft summary statistics for inference of high-dimensional data sets, since the lower-dimensional representation is computed simultaneously with learning the likelihood and without additional computational overhead. We evaluate SSNL on a wide variety of experiments, including two challenging real-world examples from the astrophysics and neuroscience literatures, and show that it either outperforms or is on par with state-of-the-art methods, making it an excellent off-the-shelf estimator for SBI for high-dimensional data sets.

1 INTRODUCTION

In the natural sciences, especially in disciplines such as biology and physics, Bayesian inference is becoming increasingly popular due to its ability both to quantify uncertainty in parameter values and to incorporate prior knowledge about quantities of interest. Bayesian statistics infers the posterior distribution $p(\theta|y) \propto p(y|\theta)p(\theta)$ of statistical parameters θ by conditioning a prior distributions $p(\theta)$ on data y. If the likelihood function $p(y|\theta)$ is available, i.e., tractable to compute, conventional Bayesian inference using Markov chain Monte Carlo or variational methods can be used for parameter inference (Brooks et al., 2011; Wainwright and Jordan, 2008). However, for many scientific hypotheses, the likelihood is not easy to compute and the experimenter merely has access to a simulator function $sim(\theta)$ that can generate synthetic data conditionally on a parameter configuration θ.

In the latter case, an emergent family of methods collectively called *simulation-based inference* (SBI, Cranmer et al. (2020)) has been proposed. Traditionally, approximate Bayesian computation (ABC, Sisson et al. (2018)), and most successfully sequential Monte Carlo ABC (SMC-ABC; e.g., Beaumont et al. (2009); Lenormand et al. (2013)) or simulated annealing ABC (SABC; e.g., Albert et al. (2015)), has been used to infer approximate posterior distributions (Pritchard et al., 1999; Ratmann et al., 2007). More recently, methods that are based on neural density or density-ratio estimation have found increased application in the natural sciences due to their reduced computational cost and convincing inferential accuracy (Brehmer et al., 2018; Delaunoy et al., 2020; Gonçalves et al., 2020; Hermans et al., 2021; Brehmer, 2021; Dax et al., 2021). Among these, several branches of methods exist. Likelihood-based methods (Papamakarios et al., 2019; Glöckler et al., 2022) fit a surrogate model for the likelihood function using neural density estimators (Papamakarios et al., 2021) which allows to do conventional Bayesian inference and which has been shown to bring significant performance advantages in comparison to ABC methods with the same computational budget. Cranmer et al. (2015); Durkan et al. (2020); Hermans et al. (2020); Thomas et al. (2022); Delaunoy et al. (2022); Miller et al. (2022) developed similar methods that instead target the likelihood-to-evidence ratio rather than the likelihood, while Papamakarios and Murray (2016); Lueckmann et al. (2017); Greenberg et al. (2019); Deistler et al. (2022); Wildberger et al. (2023); Sharrock et al. (2024) developed methods that try to approximate the posterior distribution

directly.

In the case of likelihood-based methods, the accuracy of posterior inferences might suffer due to the inability of neural density estimators to correctly approximate the surrogate likelihoods, e.g., if the data are very high-dimensional or the data are embedded in a low-dimensional manifold but lie in a higher-dimensional ambient space (Fefferman et al., 2016; Kingma and Dhariwal, 2018; Greenberg et al., 2019; Cunningham et al., 2020; Dai and Seljak, 2021; Klein et al., 2021).

To overcome this limitation, we present a new method for simulation-based inference which we call *Surjective Sequential Neural Likelihood* (SSNL) estimation. SSNL uses a surjective dimensionality-reducing normalizing flow to model the surrogate likelihood of a Bayesian model by that allowing improved density estimation and consequently improved posterior inferences. We evaluate SSNL on multiple experiments from the SBI, astrophysics and neuroscience literatures and demonstrate that it achieves superior performance in comparison to state-of-the-art methods. Conversely, we also demonstrate negative examples when our method should, in theory and empirically, not have a performance gain.

2 BACKGROUND

Given prior parameter values $\theta \sim p(\theta)$, a simulator function $sim(\theta)$ is a computer program or experimental procedure that can simulate an observation $y \leftarrow sim(\theta)$. Apart from stochasticity produced by $p(\theta)$, the simulator might be making use of another source of endogenous randomness. The simulator defines, albeit implicitly, a conditional probability distribution $p(y|\theta)$ to which the modeller does not have access or which they cannot evaluate in reasonable time. The goal of SBI is to infer the posterior distribution $p(\theta|y) \propto p(y|\theta)p(\theta)$ using synthetic data $\{(y_n, \theta_n)\}_{n=1}^N$ generated from the prior model $p(\theta)$ and simulator $sim(\theta)$. Typically, the total simulation budget N is limited and the posterior for a specific observation y_{obs} is the target of inference (Cranmer et al., 2020). In the following, we introduce relevant background on density estimation with normalizing flows and neural likelihood methods (background on neural posterior and ratio estimation methods can be found in Appendix A).

2.1 DENSITY ESTIMATION USING NORMALIZING FLOWS

Sequential density-based SBI methods (e.g., Greenberg et al. (2019); Papamakarios et al. (2019); Deistler et al. (2022)) use conditional normalizing flows to fit a surrogate model to either approximate the intractable likelihood or posterior. Normalizing flows (NFs, Papamakarios et al. (2021)) model a probability distribution via a pushforward measure as

$$q_f(y|\theta) = p\left(z_0\right) \prod_k^K \left|\det J_k\right|^{-1} \tag{1}$$

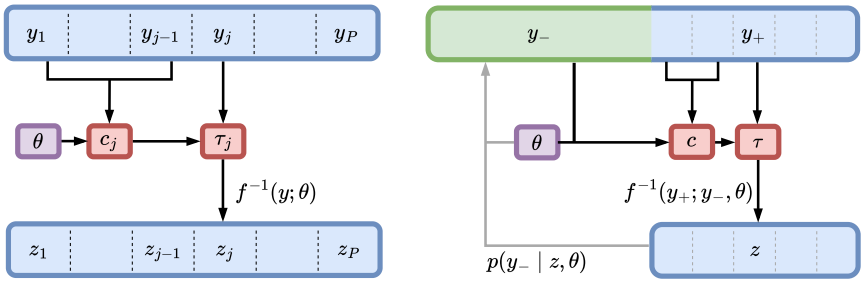
where $\det J_k = \det \frac{\partial f_k}{\partial z_{k-1}}$ is the determinant of the Jacobian matrix of a forward transformation f_k which is typically parameterized with a neural network, and $p\left(z_0\right)$ is some base distribution that has a density that can be evaluated exactly, for instance, a spherical multivariate Gaussian. The forward transformations $f = (f_1, \ldots, f_K)$ are a sequence of K diffeomorphisms which are applied consecutively to compute $y = z_K = f_K \circ \cdots \circ f_2 \circ f_1(z_0)$. The two densities q and p are related by the multiplicative terms $\det J_k$ which are needed to account for the change-of-volume induced by f_k and which are termed likelihood contribution in Nielsen et al. (2020) and Klein et al. (2021). The diffeomorphisms f_k are required to be dimensionality-preserving and invertible to be able to both evaluate the probability of a data point and to draw samples. Particularly, in autoregressive flows (Kingma et al., 2016; Papamakarios et al., 2017; Germain et al., 2015) each transformation f_k admits a Jacobian determinant which is efficient to compute and which can be decomposed into an autoregressive *conditioner* c_i and an invertible *transformer* τ_i as

$$z_{k,i} = \tau_i(z_{k-1,i}, c_i(z_{k-1,<i}))$$

where all c_i can be computed jointly using a masked neural network (Figure 1a).

2.2 NEURAL LIKELIHOOD ESTIMATION

Sequential neural likelihood estimation (SNL, Papamakarios et al. (2019)) iteratively fits a density estimator to approximate the likelihood via $q_f(y|\theta) \approx p(y|\theta)$. SNL proceeds in R rounds distributing the total simulation budget N evenly in each of these: in the first round, $r = 1$, a prior sample $\theta_n \sim p(\theta)$ of size $N_R = N/R$ is drawn and used to simulate data points $y_n \leftarrow sim(\theta_n)$ yielding the data set $\mathcal{D} = \{(y_n, \theta_n)\}_{1\ldots N_R}^r$. The simulated data is used to train a conditional normalizing flow by maximizing the expected probability $\mathbb{E}_{\mathcal{D}}\left[q_f(y|\theta)\right]$. Having access to a likelihood approximation, posterior realizations can be generated either by sampling from $\hat{p}^r(\theta|y_{\text{obs}}) \propto q_f(y_{\text{obs}}|\theta)p(\theta)$ via Markov chain Monte Carlo or via optimization by fitting a variational approximation to the approximate posterior. SNL then uses the surrogate posterior as proposal prior distribution for the next round, $r + 1$, i.e., it draws a new sample of parameters $\theta_n \sim \hat{p}^r(\theta|y_{\text{obs}})$ which are then used to simulate a new batch of pairs $\{(y_n, \theta_n)\}_{1\ldots N_R}^{r+1}$. The data sets from the previous rounds and the current round are then appended together and a new model is trained on the entire data set. With an infinite simulation budget and a sufficiently flexible density estimator q_f, SNL converges to the desired



(a) Bijection. (b) Surjection.

Figure 1: Conditional bijective and surjective flow layers illustrated with masked autoregressive flows. (a) A bijective flow layer $f^{-1}(y;\theta)$ transforms an input y_j conditional on all previous values $y_{<j}$ and a parameter vector θ using a conditioner c and transformer τ. (b) The surjective flow layer $f^{-1}(y_+;y_-,\theta)$ first splits the vector into two components y_+ and y_- and then uses the component y_- as additional conditioning variable. The implementations for conditioner and transformer remain the same as for the bijection. To evaluate the likelihood, a surjective layer additionally computes the conditional density $p(y_-|z,\theta)$ (which computationally is done after the transform).

posterior distribution $p(\theta|y_{\text{obs}})$. The simulation budget is in practice typically limited, the number simulations should be held as small as possible, and finite data leads to inaccurate approximations to the likelihood functions.

3 SURJECTIVE SEQUENTIAL NEURAL LIKELIHOOD ESTIMATION

Surjective Sequential Neural Likelihood (SSNL) estimation approximates the intractable likelihood $p(y|\theta)$ function of a Bayesian model $p(\theta|y)$ while simultaneously embedding the data in a lower-dimensional space using dimensionality-reducing surjective flows. We assume that if the data lie in a high-dimensional ambient space, which is the case for many real-world data sets like time series data, embedding them in a lower-dimensional space should improve likelihood estimation and consequently posterior inference.

We motivate the derivation of the surjective flow layer using the holistic generative framework of Nielsen et al. (2020) which models the log-probability of a P-dimensional data point y as

$$\log p(y) \simeq \log p\left(z\right) + V(y,z) + E(y,z), \quad z \sim q(z|y)$$

where $q(z|y)$ is some amortized (variational) distribution, z is a latent variable with distribution $p(z)$, $V(y,z) = \log \frac{p(y|z)}{q(z|y)}$ is denoted *likelihood contribution term* and $E(y,z) = \log \frac{q(z|y)}{p(z|y)}$ is a *bound looseness term*. Intriguingly, for inference surjections, i.e., the kind of flow layers we are considering here, the likelihood contribution can be calculated as

$$V(y,z) = \lim_{q(z|y)\to\delta(z-h^{-1}(y))} \mathbb{E}_{q(z|y)}\left[\log \frac{p(y|z)}{q(z|y)}\right]$$

where $p(y|z)$ is a conditional density, $h^{-1}: \mathcal{Y} \to \mathcal{Z}$ is a dimensionality-reducing mapping, and where we for convenience of notation denote with $h: \mathcal{Z} \to \mathcal{Y}$ a right inverse function to h^{-1} such that $h^{-1} \circ h = Id_{\mathcal{Z}}$. Critically, for surjective normalizing flows, the bound looseness equals $E(y,z) = 0$ if a right inverse function h exists.

We design a conditional surjection layer for dimensionality-reduction as follows (Figure 1b for a graphical overview). We first split the data vector $y \in \mathbb{R}^P$ into two subvectors $y = [y_-, y_+]^T$ where $y_+ \in \mathbb{R}^Q$ and Q is a hyperparameter. The subvectors are obtained by (arbitrarily) defining two disjoint permutations $\pi_+ \cup \pi_- = \{1,\dots,P\}$, $\pi_+ \cap \pi_- = \emptyset$, and then setting $y_+ = [y_{\pi_+(1)},\dots,y_{\pi_+(Q)}]^T$ and $y_- = [y_{\pi_-(1)},\dots,y_{\pi_-(P-Q)}]^T$. We then construct a conditional normalizing flow $f(z;y_-,\theta)$ (i.e., conditional on y_- and θ) and its inverse $f^{-1}(y_+;y_-,\theta)$ and define

$$\begin{aligned} q(z|y) &= \delta\left(z - f^{-1}(y_+;y_-,\theta)\right) \\ &= \delta\left(y_+ - f(z;y_-,\theta)\right) \left|\det J^{-1}\right|^{-1} \end{aligned}$$

where

$$J^{-1} = \left.\frac{\partial f^{-1}(y_+;y_-,\theta)}{\partial y_+}\right|_{y_+=f(z;y_-,\theta)}$$

is the Jacobian of the inverse mapping (see Appendix B for details). Using this result and the conditional distribution $p(y|z) = p(y_-|z,\theta)$ the likelihood contribution for a surjection layer becomes

$$\begin{aligned} V(y,z) &= \lim_{q(z|y)\to\delta(z-h^{-1}(y))} \mathbb{E}_{q(z|y)}\left[\log \frac{p(y|z)}{q(z|y)}\right] \\ &= \int \delta\left(z - f^{-1}(y_+;y_-,\theta)\right) \\ &\qquad \log \frac{p(y_-|z,\theta)}{\delta\left(z - f^{-1}(y_+;y_-,\theta)\right)} \mathrm{d}z \\ &= \log p\left(y_-|f^{-1}(y_+;y_-,\theta)\right) - \log\left|\det J^{-1}\right|^{-1} \end{aligned}$$

where we used the change of variables $\tilde{y}^+ = f(z; y_-, \theta)$ yielding $\mathrm{d}\tilde{y}^+ = \mathrm{d}z|\det J^{-1}|^{-1}$. The likelihood of an observation using a surjective flow is consequently the product of three terms:

$$p\left(z\right) p(y_-|z,\theta)\left|\det J\right|^{-1}$$

where $p(z)$ is a base distribution, $z = f^{-1}(y_+; y_-, \theta)$ and $\det J = \det \frac{\partial f(\cdot; y_-, \theta)}{\partial z}$ is again the Jacobian determinant of the forward transformation acting on the lower-dimensional vector z (see Appendix B for a detailed derivation of the surjection layer). Note that this representation strictly extends the one by Klein et al. (2021), since here we construct flows that are conditioned on the parameter vector θ. Analogously to multi-layered bijective flows (Equation (1)), the conditional density of a normalizing flow that consists of K dimensionality-reducing layers has the following form:

$$q_f(y|\theta) = p\left(z_0\right) \prod_k^K p(z_{k,-}|f_k^{-1}(z_{k,+}; z_{k,-}, \theta))\left|\det J_k\right|^{-1}$$

where $z_{k,-}$ and $z_{k,+}$ are subvectors of z_k that have been constructed as above and $J_k = \frac{\partial f_k(\cdot; z_{k,-}, \theta)}{\partial z_{k-1,+}}$ is the Jacobian of the kth surjective transformation f_k.

For simulation-based-inference, we model the likelihood estimator $q_f(y|\theta)$ as a composition of dimensionality-preserving and -reducing layers:

$$q_f(y|\theta) = p\left(z_0\right) \prod_{k \in \mathcal{K}_{\text{pres}}} \left|\det J_k\right|^{-1} \prod_{k \in \mathcal{K}_{\text{red}}} p(z_{k,-}|f_k^{-1}(z_{k,+}; z_{k,-}, \theta))\left|\det J_k\right|^{-1} \tag{2}$$

where $\mathcal{K}_{\text{pres}}$ and $\mathcal{K}_{\text{red}}$ represent sets of indexes for dimensionality-preserving and -reducing flow layers, respectively. For instance, for a total of $K = 5$ normalizing flow layers, one could alternate between bijections and surjections by setting the sets $\mathcal{K}_{\text{pres}} = \{1, 3, 5\}$ and $\mathcal{K}_{\text{red}} = \{2, 4\}$. Here, we parameterize f using masked autoregressive flows but in general any flow architecture, such as coupling flows (Dinh et al., 2015, 2017), neural spline flows (Durkan et al., 2019) or neural autoregressive flows (Huang et al., 2018; De Cao et al., 2020), is possible.

The dimensionality-reducing flow is fully deterministic in the pullback direction, i.e., in the case of likelihood estimation, but requires sampling from the conditional $p\left(y_-|z,\theta\right)$ during the forward transformation and, hence, has additional stochastic components other than the base distribution $p(z_0)$. For our setting, i.e., density estimation, this is however not a limitation.

The lower-dimensional embedding of SSNL solves previous issues of neural likelihood methods when scaled to high-dimensional data sets. In addition, through the dimensionality-reduction the flows require less trainable parameters, which can speed up computation such that more of the computational budget can be used for the simulator or more expressive architectures. The embedding acts, albeit only conceptually, as a collection of summary statistics which consequently replaces the need of manually defining them.

Algorithm 1 Surjective sequential neural likelihood

Inputs: observation y_{obs}, prior distribution $p(\theta)$, surjective normalizing flow $q_f(y|\theta)$, simulations per round N_R, number of rounds R
Outputs: approximate posterior distribution $\hat{p}^R(\theta|y_{\text{obs}})$
Initialize proposal $\hat{p}^0(\theta|y_{\text{obs}}) \leftarrow p(\theta)$, data set $\mathcal{D} = \{\}$
for $r \leftarrow 1, \ldots, R$ **do**
  for $n \leftarrow 1, \ldots, N_R$ **do**
    Sample $\theta_n \sim \hat{p}^{r-1}(\theta|y_{\text{obs}})$
    Simulate $y_n \leftarrow sim(\theta_n)$ using the simulator function
    Concatenate $\mathcal{D} \leftarrow \{\mathcal{D}, (y_n, \theta_n)\}$
  end for
  Train $q_f(y|\theta)$ on $\mathcal{D}$
  Set $\hat{p}^r(\theta|y_{\text{obs}}) \propto q_f(y_{\text{obs}}|\theta)p(\theta)$
end for

Like other sequential methods, SSNL is trained in R rounds where in every round a new proposal posterior is defined. The proposal posterior can be either sampled from using MCMC methods or approximated with another conditional distribution using variational inference (see Algorithm 1).

4 EXPERIMENTS

We compare SSNL to Sequential Neural Likelihood (SNL, Papamakarios et al. (2019)), Sequential Neural Posterior Estimation-C (SNPE-C, Greenberg et al. (2019)), Sequential Neural Ratio Estimation-C (SNRE-C, Miller et al. (2022)), Sequential Neural Approximate Sufficient Statistics (SNASS, Chen et al. (2021) and Sequential Neural Approximate Slice Sufficient Statistics (SNASS, Chen et al. (2023)) on seven synthetic experiments, a solar dynamo model from the astrophysics literature and a neural mass model from neuroscience to highlight the advantages and disadvantages of the method. We chose SNPE-C as neural posterior method since we found it is still state-of-the-art or at least highly competitive on a large number of experimental benchmarks (see, e.g., Deistler et al. (2022); Wildberger et al. (2023)). Similarly, SNRE-C is to our knowledge state-of-the-art among methods for neural ratio estimation. Conceptually related to our method, SNASS and SNASSS first compute a set of near-sufficient summary statistics using embedding networks and then use SNL to fit a posterior approximation.

We followed the experimental details of Papamakarios et al.

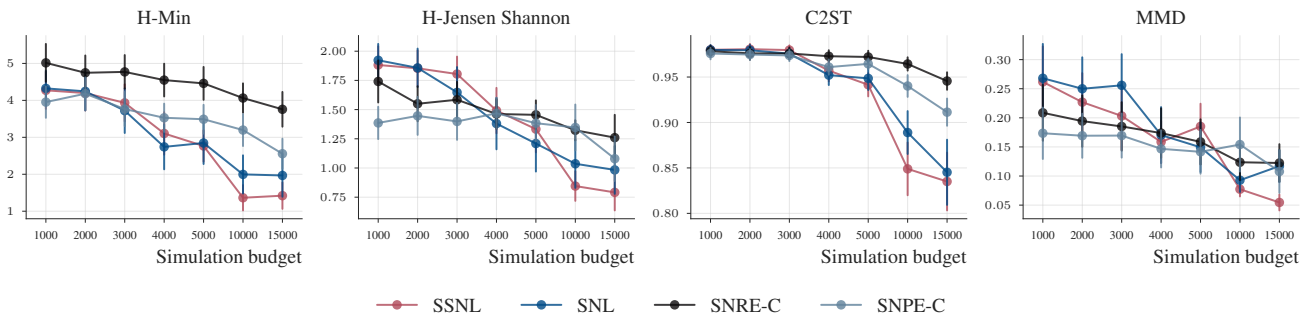


Figure 2: Method accuracies on SLCP with different evaluation measures (lower is better; y-axis shows measure).

(2019), Greenberg et al. (2019) and Miller et al. (2022). In short, for SSNL, SNL and SNPE-C we use masked autoregressive flows (MAFs) with five flow layers where each layer uses a neural network with two hidden layers and 50 nodes per layer. Since SNASS and SNASSS have to fit additional summary and critic networks, these use MAFs with three and two layers, respectively, to have roughly the same number of parameters as the previous methods. SSNL uses a dimensionality-reducing surjection in the middle layer for which the conditional density $p\left(z_k^- | f_k^{-1}(z_k^+; z_k^-, \theta), \theta\right)$ is parameterized using an MLP with two layers of 50 nodes each. We selected the third layer as surjection such that the entire data set is "processed" once in each direction before reducing the dimensionality of the data. We evaluated surjection layers that reduce the dimensionality by 25%, 50% or 75%, respectively (see Appendix D for all experimental details).

For each experimental model, we sample a vector of true parameters $\theta_{\text{obs}} \sim p(\theta)$ and then simulate an observation $y_{\text{obs}} \leftarrow sim(\theta_{\text{obs}})$ which is then used to approximate $p(\theta|y_{\text{obs}})$. We repeat this data generating process for 10 different seeds. We evaluate each method sequentially in $R = 15$ rounds using a total simulation budget of $N = 15\,000$: in each round r we draw a sample θ_n^r of size $N_R = 1\,000$ from the trained surrogate posterior (or prior if in the first round, respectively), simulate observations $y_n^r \leftarrow sim(\theta_n^r)$, and train the density estimator/classifier on all available data (i.e., including the data from all previous rounds, yielding a simulation budget of 1 000 for the first round, 2 000 for the second round, etc.). After training, we compare samples from the posterior approximation of a method of each round to samples obtained from MCMC and compute divergence measures between the two samples. For the solar dynamo and neural mass models, we compare the surrogate posterior samples to the true parameter values θ_{obs} as in prior work (e.g., Rodrigues et al. (2021) or Buckwar et al. (2020)).

4.1 COMPARING POSTERIOR DISTRIBUTIONS

Previous work has evaluated the accuracy of the approximated posteriors to the true posterior (or rather the posterior obtained via Monte Carlo samples), mainly using maximum mean discrepancy (MMD; Gretton et al. (2012); Sutherland et al. (2017)) and classifier two-sample tests (C2ST; Lopez-Paz and Oquab (2017)). Recently, Zhao et al. (2022) introduced a general H-divergence to assess the similarity of two (empirical) distributions, p and q, and demonstrated that their method has higher power than members of the MMD and C2ST families in several experimental evaluations while having a low number of hyperparameters to optimize. Specifically, Zhao et al. (2022) propose to use the divergence

$$D_\ell^\phi(p||q) = \phi\left(H_\ell\left(\tfrac{p+q}{2}\right) - H_\ell(p), H_\ell\left(\tfrac{p+q}{2}\right) - H_\ell(q)\right)$$

where the H-min divergence $D_\ell^{\text{Min}} = H_\ell\left(\frac{p+q}{2}\right) - \min(H_\ell(p), H_\ell(q))$ and H-Jensen Shannon divergence $D_\ell^{\text{JS}} = H_\ell\left(\frac{p+q}{2}\right) - \frac{1}{2}(H_\ell(p), H_\ell(q))$ are special cases. $H_\ell(p) = \inf_{a\in\mathcal{A}} \mathbb{E}_p[\ell(X, a)]$ is the Bayes optimal loss of some decision function over an action set $\mathcal{A}$ and the loss ℓ can in practice be implemented, e.g., using the negative log-likelihood of a density estimator such as kernel density estimator or Gaussian mixture model (see Appendix D for details on H-divergences).

We evaluated H-Min and H-Jensen Shannon divergences on the notorious simple-likelihood-complex-posterior model (SLCP; see Appendix E for a description) following the experimental details in Zhao et al. (2022) and compared them to MMD and C2ST distances (Figure 2). We found that the profiles of H-Min or H-Jensen Shannon have similar trends as C2ST and MMD, respectively (the H-Jensen Shannon divergence is in fact strictly larger than the family of MMD distances (Zhao et al., 2022)).

Hence, we propose to use both the H-Min and H-Jensen Shannon divergences as model evaluation metrics for SBI benchmarks due to their power, implementational simplicity and low number of tunable hyperparameters, and will report them for the experimental evaluations. Note that in the SLCP

example, which we used to assess the different divergences, SSNL outperforms the three baselines consistently with a sufficient simulation budget.

4.2 SBI MODEL BENCHMARKS

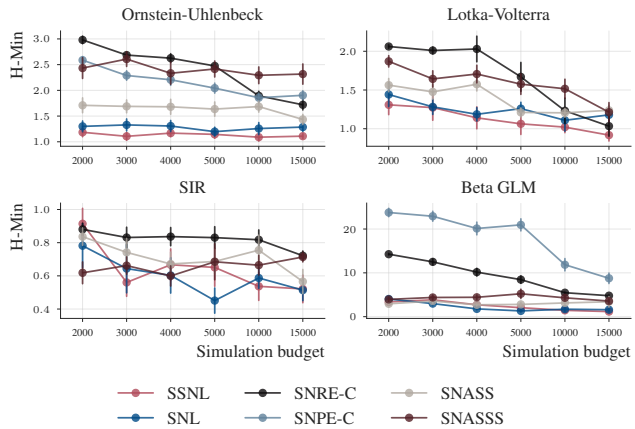
We first evaluate SSNL on multiple benchmark models from the SBI literature (i.e., Ornstein-Uhlenbeck, Lotka-Volterra, SIR, and generalized linear model (GLM)) and discuss when it should have performance benefits over alternative methods. We then demonstrate using two negative examples where SSNL breaks and where it should fail to outperform the baselines (Gaussian mixture model and hyperboloid model). For a detailed description of the six experimental models which we omit here, we refer to Appendix E.

Results For SSNL, we first determined the optimal embedding dimensionality in the following way: we extract the validation loss, i.e., the negative log-likelihood on the validation set, after training and use the embedding dimensionality corresponding to the network that achieved the lowest validation loss (Figure 3b)[1]. Since the loss profiles on all four experiments are roughly the same for each parameterization, we, for simplicity, chose to use the networks that reduce the dimensionality to 75% for each experimental model.

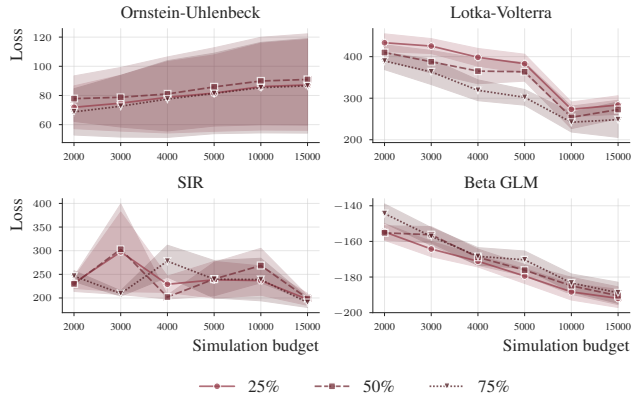
For the two time series models Ornstein-Uhlenbeck (OU) and Lotka-Volterra (LV), SSNL consistently outperforms all baselines. SSNL is on par with SNL on the SIR and Beta GLM models (see Figure 3a). The SIR model is the only case with mixed, inconsistent results where for different simulation budgets SNL gets outperformed by SSNL or vice versa (the figures do not show SNPE-C for LV and SIR due to its bad performance, see Figure 9 in the Appendix for complete results).

Autocorrelation and intrinsic dimensionality We assessed in which case and why SSNL has a performance advantage over SNL and argue that a combination of autocorrelation and intrinsic dimensionality (ID) of a data set might be indicative of it (see Figure 3c where realizations of a time series model with different parameter values are shown). Notably, for the Ornstein-Uhlenbeck and Lotka-Volterra models the autocorrelation pattern seems to benefit dimensionality-reducing methods. For the Ornstein-Uhlenbeck process, the autocorrelation converges to zero

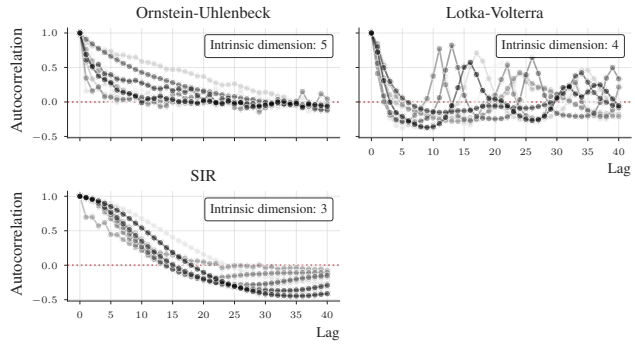
[1]This could be done more rigorously, e.g., by splitting the data into an additional test set and evaluating its loss or by simply reducing the dimensionality sequentially such that the embedding has as many dimensions as required summary statistics, but we found this simple heuristic to be sufficient and it does not require additional computation. Furthermore, since the data is simulated with iid noise, the likelihoods on validation and test sets should be almost equal. Alternatively, information-theoretical approaches could also be applied.



(a) Posterior divergences.



(b) Likelihood profiles.



(c) Autocorrelations and intrinsic dimensions.

Figure 3: Experimental results of OU, LV, SIR and GLM models. (a) H-Min divergences of all models plotted against the size of simulated data (lower is better). (b) Validation likelihood profiles of SSNL models when the middle layer reduces the dimensionality by 25%, 50% or 75%, respectively. The performances are similar for all models which is why we used the most conservative reduction, i.e. 75% for all models. (c) Autocorrelation (AR) plots for the three time series models up to a lag of 40 (black shades correspond to realizations of a time series with different parameter values). The AR for the OU model converges to zero while the AR for the LV has a self-repeating structure. The SIR model has a single saddle-point and does not converge.

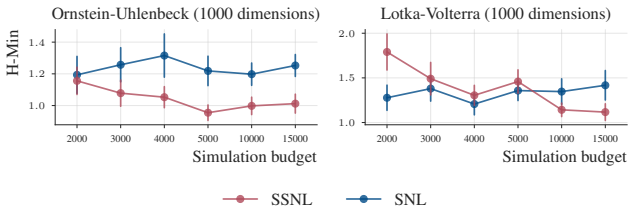


Figure 4: Posterior divergences on 1000-dimensional Ornstein-Uhlenbeck and Lotka-Volterra models.

when considering longer lags meaning that information beyond a certain point is not informative of the parameters any more. Similarly, for the Lotka-Volterra process the autocorrelation patterns are repetitive after a certain lag meaning that the data at larger time points is basically a copy of previous time points (compare Figure 7 in the Appendix). In the case of the SIR model, the autocorrelation has first an negative slope and then changes the sign of its gradient function after reaching a saddle-point. Consequently, the entire time-series is informative of the parameters and dimensionality reduction has supposedly only little advantage over dimension-preservation (more experimental results can be found in Appendix F).

Increasing the dimensionality The benefit of dimension reduction depends arguably on the length of the time series and its signal-to-noise ratio. To validate this hypothesis, we replicated the OU and LV experiments but increased the number of time points from 100 to 1000 and observed that the performance difference between SSNL and SNL in fact increases. For Ornstein-Uhlenbeck, SSNL has a significant performance advantage over SNL, while for Lotka-Volterra the same can be observed with sufficient simulation budget (Figure 4). We hypothesize that this is due to the fact that in some cases learning the high-dimensional conditional density $p\left(y_- | f^{-1}(y_+; y_-, \theta)\right)$ requires an increased sample size.

Negative examples and limitations The performance of SSNL depends on whether the parameter-related information in the data can be represented in a lower-dimensional space. In scenarios where this is not the case, e.g., on Gaussian mixture or hyperboloid models, SNPE-C or SNL expectedly outperform SSNL (see Figure 10 in the Appendix).

4.3 SOLAR DYNAMO

We applied SSNL to a real-world solar dynamo model from the solar physics literature that models the magnetic field strength of the sun (see Charbonneau et al. (2005) and references therein). The model is a non-linear time series model with both additive and multiplicative noise terms

$$g(y) = \tfrac{1}{2}[1 + \mathrm{erf}(\tfrac{y-b_1}{w_1})][1 - \mathrm{erf}(\tfrac{y-b_2}{w_2})]$$
$$\alpha_t \sim \mathcal{U}(\theta_1, \theta_1 + \theta_2), \quad \epsilon_t \sim \mathcal{U}(0, \theta_3),$$
$$y_{t+1} \leftarrow \alpha_t g(y_t) y_t + \epsilon_t$$

The model is interesting, because it has more noise components than observed outcomes and integrating out the noise components yields a marginal likelihood that is outside the exponential family. Consequently, the number of sufficient statistics for such a model is unbounded (with the length of the time series T) according to Pitman-Koopman-Darmois theorem. We choose the prior $p(\theta)$ and hyperparameters b and w as in Albert et al. (2022) and simulate a single time series of length $T = 100$ (see Appendix E.8 for details).

SSNL consistently outperforms the five baselines for this experiment (Figure 5a left column). Having a closer look at the posterior distributions of one experimental run, one can observe that SSNL already after the first round recovers the true parameters reliably while the posterior mean of SNL is heavily biased (Figure 6). After the final round, both methods converge to the true parameter values.

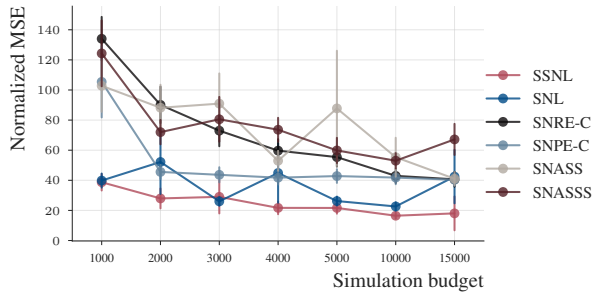
4.4 NEURAL MASS MODEL

We also evaluate SSNL on the stochastic version of the Jansen-Rit neural mass model (Ableidinger et al., 2017) which describes the collective electrical activity of neurons by modelling interactions of cells (see Ableidinger et al. (2017); Buckwar et al. (2020); Rodrigues et al. (2021) for details). The model is a 6-dimensional SDE of the form

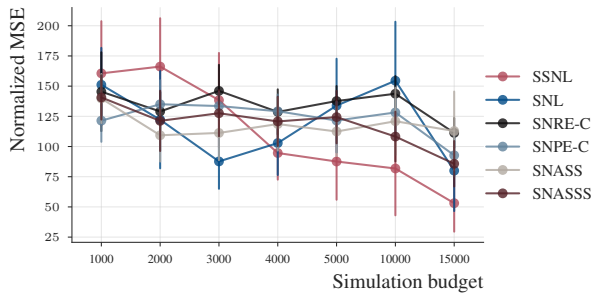
$$\mathrm{d}\begin{pmatrix} R_t \\ S_t \end{pmatrix} = \begin{pmatrix} S_t \\ -\Gamma^2 R_t - 2\Gamma S_t + G_\theta(R_t) \end{pmatrix} \mathrm{d}t + \begin{pmatrix} 0 \\ \Sigma_\theta \end{pmatrix} \mathrm{d}W_t$$

where $R = [Y_1, Y_2, Y_3]^T$, $S = [Y_4, Y_5, Y_6]^T$, W_t is a Wiener process, Σ_θ is a diagonal covariance matrix, G_θ is a displacement vector, Γ is a matrix, and θ is a four-dimensional random vector with uniform prior (see Appendix E.9 for details).

With a sufficient simulation budget, in this case 4000 simulations, SSNL convincingly outperforms the baselines having the lowest MSE. As before (see, e.g., Figure 4), we hypothesize that the performance of SSNL is worse for lower simulation budgets, since an additional conditional density has to be learned. Intriguingly, the inferences of SNL, which like SSNL approximates the likelihood function, are significantly worse than for SSNL and overall inconsistent, indicating that SSNL is in general an excellent off-the-shelf estimator for high-dimensional data sets (more experimental results can be found in Appendix F).



(a) Solar dynamo



(b) Neural mass model

Figure 5: Solar dynamo and neural mass model evaluation (we show the MSE w.r.t. the prior sample θ_{obs} that was used to simulate the observation y_{obs}. Values are normalized by the minimum MSE in the entire data set).

5 RELATED WORK

Neural simulation-based inference is an emergent field with significant advances in the recent past. In addition to (sequential) neural approximations to the posterior (Papamakarios and Murray, 2016; Lueckmann et al., 2017; Greenberg et al., 2019; Deistler et al., 2022; Wildberger et al., 2023), likelihood (Papamakarios et al., 2019; Glöckler et al., 2022) or likelihood-ratio (Cranmer et al., 2015; Durkan et al., 2020; Hermans et al., 2020; Thomas et al., 2022; Delaunoy et al., 2022; Miller et al., 2022), recent approaches have utilized flow matching and score-based models for posterior inference (Schmitt et al., 2024; Geffner et al., 2023; Wildberger et al., 2023), albeit mainly in a non-sequential manner. Score-based NPE methods are a fruitful research direction which do not restrict the architecture of the neural network architecture. More recently, Gruner et al. (2023) proposed a new method that is targeted at models in which the posterior is conditioned on multiple observations simultaneously. Jia (2024) introduce a new family of methods, called *neural quantile estimation*, which uses quantile regression to either approximate the intractable posterior or likelihood functions of a model. Glaser et al. (2022); Pacchiardi and Dutta (2022) propose approaches to SBI using energy-based models as density estimators. Finally, Yao et al. (2024) propose a method based on stacking to combine the results of multiple posterior inferences, i.e., when multiple posterior approximations from different methods are available.

Related to our work, Alsing and Wandelt (2018); Chen et al. (2021, 2023) have developed methods for SBI to compute summary statistics which, however, requires learning an additional embedding network while our approach learns an embedding and computes likelihood approximations in a single step and without computational overhead. Radev et al. (2023) discuss an approach that learns three networks for posterior approximations, likelihood approximations and summary statistic computation. Beck et al. (2022) discuss marginalization of data dimensions to see the influence of different covariates on the posterior distribution and propose an approach to find informative data dimensions.

In order to evaluate inferences of non-sequential procedures of models for which samples from the true posterior are not available, Linhart et al. (2023) have proposed a local procedure based on classifier tests. Similarly, Yao and Domke (2023) proposed a diagnostic related to, and with higher power than, simulation-based calibration (Talts et al. (2018); unfortunately, for neural likelihood methods we found them computationally infeasible to use, since they require either computing a vast number of permutation tests of which each requires learning a classifier or repeatedly sampling from the surrogate posterior for every model).

6 CONCLUSION

We introduced *Surjective Sequential Neural Likelihood* estimation, a new neural likelihood method for simulation-based inference for high-dimensional data. SSNL uses dimensionality-reducing surjections to embed the data in a lower-dimensional space while simultaneously learning the likelihood function.

SSNL performs particularly well when applied to high-dimensional time series data outperforming state-of-the-art methods and is on par with them in other experiments, making it an excellent off-the-shelf estimator for high-dimensional data sets. As a limitation, we identified that due to introducing an additional conditional density, SSNL in some cases requires a higher sample size than other flow-based models like SNL or SNPE-C.

We anticipate that our method will complement other approaches in SBI well, but do not argue that it should be generally preferred over others. For instance, when posteriors have a simple geometry and the likelihood function is complicated to approximate, posterior estimators like SNPE-C often provide superior performance. Similarly, neural ratio methods, such as SNRE-C, are typically easier to fit as flow-based methods, since they only require training a classifier.

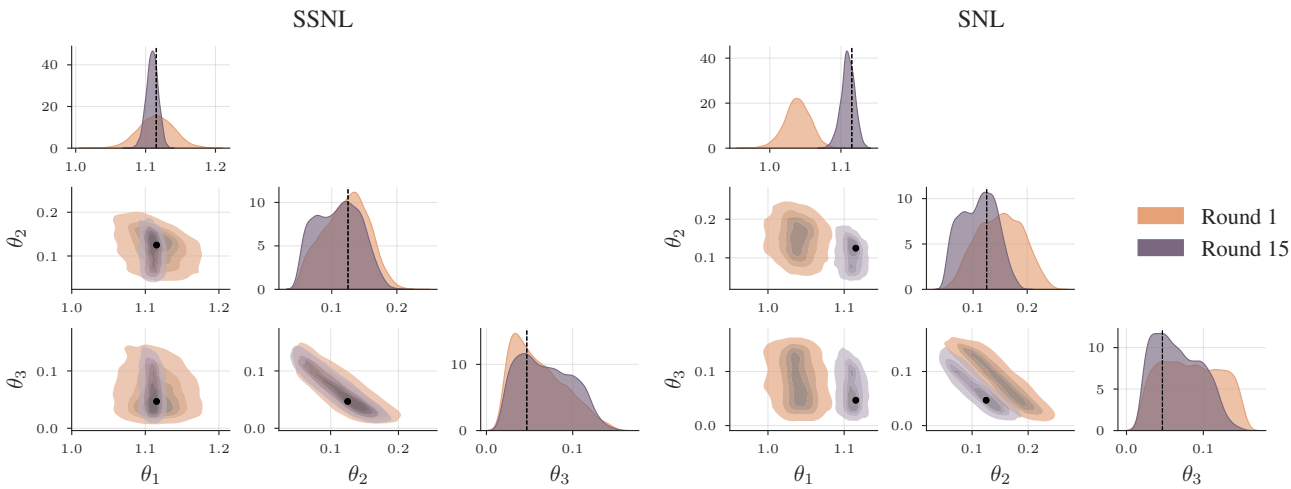


Figure 6: Solar dynamo posterior distributions of SSNL and SNL after the 1st and 15th round (shown as kernel density estimates. Black dots and lines represent true parameter values).

Future research could elucidate how deeper (stacked) surjective architectures impact posterior inference and if more suitable surjective architectures can be developed for SBI. The application of information-theoretic criteria to more rigorously determine the dimensionality of the latent space, for instance, as done in Chen et al. (2023), is another possible research direction.

Acknowledgements

This research was supported by the Swiss National Science Foundation (Grant No. 200021_208249). We thank all anonymous reviewers for their feedback which helped to improve the manuscript.

References

Ableidinger, M., Buckwar, E., and Hinterleitner, H. (2017). A stochastic version of the Jansen and Rit neural mass model: analysis and numerics. *The Journal of Mathematical Neuroscience*, 7(1):1–35.

Albert, C., Künsch, H. R., and Scheidegger, A. (2015). A simulated annealing approach to approximate Bayes computations. *Statistics and Computing*, 25:1217–1232.

Albert, C., Ulzega, S., Ozdemir, F., Perez-Cruz, F., and Mira, A. (2022). Learning summary statistics for Bayesian inference with autoencoders. *SciPost Physics Core*, 5(3):043.

Alsing, J. and Wandelt, B. (2018). Generalized massive optimal data compression. *Monthly Notices of the Royal Astronomical Society: Letters*, 476(1):L60–L64.

Amsaleg, L., Chelly, O., Houle, M. E., Kawarabayashi, K.-i., Radovanović, M., and Treeratanajaru, W. (2022). Intrinsic dimensionality estimation within tight localities: A theoretical and experimental analysis. *arXiv preprint arXiv:2209.14475*.

Babuschkin, I., Baumli, et al. (2020). The DeepMind JAX Ecosystem.

Bac, J., Mirkes, E. M., Gorban, A. N., Tyukin, I., and Zinovyev, A. (2021). Scikit-dimension: a python package for intrinsic dimension estimation. *Entropy*, 23(10):1368.

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

Beck, J., Deistler, M., Bernaerts, Y., Macke, J. H., and Berens, P. (2022). Efficient identification of informative features in simulation-based inference. In *Advances in Neural Information Processing Systems*.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. (2018). JAX: composable transformations of Python+NumPy programs.

Brehmer, J. (2021). Simulation-based inference in particle physics. *Nature Reviews Physics*, 3(5):305–305.

Brehmer, J., Cranmer, K., Louppe, G., and Pavez, J. (2018). Constraining effective field theories with machine learning. *Phys. Rev. Lett.*, 121:111801.

Brooks, S., Gelman, A., Jones, G., and Meng, X.-L. (2011). *Handbook of Markov Chain Monte Carlo*. CRC press.

Buckwar, E., Tamborrino, M., and Tubikanec, I. (2020). Spectral density-based and measure-preserving ABC for partially observed diffusion processes. an illustration on

Hamiltonian SDEs. *Statistics and Computing*, 30:627–648.

Charbonneau, P., St-Jean, C., and Zacharias, P. (2005). Fluctuations in Babcock-Leighton dynamos. i. period doubling and transition to chaos. *The Astrophysical Journal*, 619(1):613.

Chen, R. T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. (2018). Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*.

Chen, Y., Gutmann, M. U., and Weller, A. (2023). Is learning summary statistics necessary for likelihood-free inference? In *Proceedings of the 40th International Conference on Machine Learning*.

Chen, Y., Zhang, D., Gutmann, M. U., Courville, A., and Zhu, Z. (2021). Neural approximate sufficient statistics for implicit models. In *International Conference on Learning Representations*.

Cranmer, K., Brehmer, J., and Louppe, G. (2020). The frontier of simulation-based inference. *Proceedings of the National Academy of Sciences*, 117(48):30055–30062.

Cranmer, K., Pavez, J., and Louppe, G. (2015). Approximating likelihood ratios with calibrated discriminative classifiers. *arXiv preprint arXiv:1506.02169*.

Cunningham, E., Zabounidis, R., Agrawal, A., Fiterau, M., and Sheldon, D. (2020). Normalizing flows across dimensions. In *ICML Workshop on Theoretical Foundations and Applications of Deep Generative Models*.

Dai, B. and Seljak, U. (2021). Sliced iterative normalizing flows. In *ICML Workshop on Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models*.

Dax, M., Green, S. R., Gair, J., Macke, J. H., Buonanno, A., and Schölkopf, B. (2021). Real-time gravitational wave science with neural posterior estimation. *Phys. Rev. Lett.*, 127:241103.

De Cao, N., Aziz, W., and Titov, I. (2020). Block neural autoregressive flow. In *Proceedings of the Thirty-Fifth Conference on Uncertainty in Artificial Intelligence*.

Deistler, M., Goncalves, P. J., and Macke, J. H. (2022). Truncated proposals for scalable and hassle-free simulation-based inference. In *Advances in Neural Information Processing Systems*.

Delaunoy, A., Hermans, J., Rozet, F., Wehenkel, A., and Louppe, G. (2022). Towards reliable simulation-based inference with balanced neural ratio estimation. In *Advances in Neural Information Processing Systems*.

Delaunoy, A., Wehenkel, A., Hinderer, T., Nissanke, S., Weniger, C., Williamson, A. R., and Louppe, G. (2020). Lightning-fast gravitational wave parameter inference through neural amortization. In *Workshop on Machine Learning and the Physical Sciences, Advances in Neural Information Processing Systems*.

Dinh, L., Krueger, D., and Bengio, Y. (2015). NICE: nonlinear independent components estimation. In *Workshop Track, International Conference on Learning Representations*.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. (2017). Density estimation using real NVP. In *International Conference on Learning Representations*.

Dirmeier, S. (2024). Surjectors: surjection layers for density estimation with normalizing flows. *Journal of Open Source Software*, 9(94):6188.

Dirmeier, S., Ulzega, S., Mira, A., and Albert, C. (2024). Simulation-based inference with the Python package sbijax. *arXiv preprint arXiv:2409.19435*.

Durkan, C., Bekasov, A., Murray, I., and Papamakarios, G. (2019). Neural spline flows. In *Advances in Neural Information Processing Systems*.

Durkan, C., Murray, I., and Papamakarios, G. (2020). On contrastive learning for likelihood-free inference. In *Proceedings of the 37th International Conference on Machine Learning*.

Fefferman, C., Mitter, S., and Narayanan, H. (2016). Testing the manifold hypothesis. *Journal of the American Mathematical Society*, 29(4):983–1049.

Ferrari, S. and Cribari-Neto, F. (2004). Beta regression for modelling rates and proportions. *Journal of Applied Statistics*, 31(7):799–815.

Forbes, F., Nguyen, H. D., Nguyen, T., and Arbel, J. (2022). Summary statistics and discrepancy measures for approximate bayesian computation via surrogate posteriors. *Statistics and Computing*, 32(5):85.

Gabry, J., Simpson, D., Vehtari, A., Betancourt, M., and Gelman, A. (2019). Visualization in bayesian workflow. *Journal of the Royal Statistical Society Series A: Statistics in Society*, 182(2):389–402.

Geffner, T., Papamakarios, G., and Mnih, A. (2023). Compositional score modeling for simulation-based inference. In *Proceedings of the 40th International Conference on Machine Learning*.

Gelman, A. and Rubin, D. B. (1992). Inference from iterative simulation using multiple sequences. *Statistical Science*, 7(4):457 – 472.

Germain, M., Gregor, K., Murray, I., and Larochelle, H. (2015). Made: Masked autoencoder for distribution estimation. In *Proceedings of the 32nd International Conference on Machine Learning*.

Glaser, P., Arbel, M., Doucet, A., and Gretton, A. (2022). Maximum likelihood learning of energy-based models for simulation-based inference. *arXiv preprint arXiv:2210.14756*.

Glöckler, M., Deistler, M., and Macke, J. H. (2022). Variational methods for simulation-based inference. In *International Conference on Learning Representations*.

Gonçalves, P. J., Lueckmann, J.-M., Deistler, M., Nonnenmacher, M., Öcal, K., Bassetto, G., Chintaluri, C., Podlaski, W. F., Haddad, S. A., Vogels, T. P., et al. (2020). Training deep neural density estimators to identify mechanistic models of neural dynamics. *Elife*, 9:e56261.

Grathwohl, W., Chen, R. T., Bettencourt, J., and Duvenaud, D. (2019). Ffjord: Free-form continuous dynamics for scalable reversible generative models. In *International Conference on Learning Representations*.

Greenberg, D., Nonnenmacher, M., and Macke, J. (2019). Automatic posterior transformation for likelihood-free inference. In *Proceedings of the 36th International Conference on Machine Learning*.

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

Gruner, T., Belousov, B., Muratore, F., Palenicek, D., and Peters, J. (2023). Pseudo-likelihood inference. In *Advances in Neural Information Processing Systems*.

Hermans, J., Banik, N., Weniger, C., Bertone, G., and Louppe, G. (2021). Towards constraining warm dark matter with stellar streams through neural simulation-based inference. *Monthly Notices of the Royal Astronomical Society*, 507(2).

Hermans, J., Begy, V., and Louppe, G. (2020). Likelihood-free MCMC with amortized approximate ratio estimators. In *Proceedings of the 37th International Conference on Machine Learning*.

Ho, J., Chen, X., Srinivas, A., Duan, Y., and Abbeel, P. (2019). Flow++: Improving flow-based generative models with variational dequantization and architecture design. In *Proceedings of the 36th International Conference on Machine Learning*.

Hoffman, M. D. and Gelman, A. (2014). The No-U-Turn sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15(47):1593–1623.

Huang, C.-W., Krueger, D., Lacoste, A., and Courville, A. (2018). Neural autoregressive flows. In *Proceedings of the 35th International Conference on Machine Learning*.

Jia, H. (2024). Simulation-based inference with quantile regression. In *Proceedings of the 41st International Conference on Machine Learning*.

Kidger, P. (2021). *On Neural Differential Equations*. PhD thesis, University of Oxford.

Kingma, D. P. and Dhariwal, P. (2018). Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*.

Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. (2016). Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*.

Klein, S., Raine, J. A., Pina-Otey, S., Voloshynovskiy, S., and Golling, T. (2021). Funnels: Exact maximum likelihood with dimensionality reduction. In *Workshop on Bayesian Deep Learning, Advances in Neural Information Processing Systems*.

Lenormand, M., Jabot, F., and Deffuant, G. (2013). Adaptive approximate Bayesian computation for complex models. *Computational Statistics*, 28(6):2777–2796.

Linhart, J., Gramfort, A., and Rodrigues, P. L. C. (2023). L-C2ST: Local diagnostics for posterior approximations in simulation-based inference. In *Advances in Neural Information Processing Systems*.

Lopez-Paz, D. and Oquab, M. (2017). Revisiting classifier two-sample tests. In *International Conference on Learning Representations*.

Lueckmann, J.-M., Boelts, J., Greenberg, D., Goncalves, P., and Macke, J. (2021). Benchmarking simulation-based inference. In *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics*.

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

Miller, B. K., Weniger, C., and Forré, P. (2022). Contrastive neural ratio estimation. In *Advances in Neural Information Processing Systems*.

Nielsen, D., Jaini, P., Hoogeboom, E., Winther, O., and Welling, M. (2020). Survae flows: Surjections to bridge the gap between vaes and flows. In *Advances in Neural Information Processing Systems*.

Pacchiardi, L. and Dutta, R. (2022). Score matched neural exponential families for likelihood-free inference. *The Journal of Machine Learning Research*, 23(1):1745–1815.

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

Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. (2021). Normalizing flows for probabilistic modeling and inference. *The Journal of Machine Learning Research*, 22(1):2617–2680.

Papamakarios, G., Pavlakou, T., and Murray, I. (2017). Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*.

Papamakarios, G., Sterratt, D., and Murray, I. (2019). Sequential neural likelihood: Fast likelihood-free inference with autoregressive flows. In *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics*.

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

Radev, S. T., Schmitt, M., Pratz, V., Picchini, U., Koethe, U., and Buerkner, P. (2023). JANA: Jointly amortized neural approximation of complex bayesian models. In *Proceedings of the Thirty-Ninth Conference on Uncertainty in Artificial Intelligence*.

Ratmann, O., Jørgensen, O., Hinkley, T., Stumpf, M., Richardson, S., and Wiuf, C. (2007). Using likelihood-free inference to compare evolutionary dynamics of the protein networks of H. pylori and P. falciparum. *PLoS Computational Biology*, 3(11):e230.

Rodrigues, P., Moreau, T., Louppe, G., and Gramfort, A. (2021). HNPE: Leveraging global parameters for neural posterior estimation. In *Advances in Neural Information Processing Systems*.

Särkkä, S. and Solin, A. (2019). *Applied stochastic differential equations*. Cambridge University Press.

Schmitt, M., Pratz, V., Köthe, U., Bürkner, P.-C., and Radev, S. T. (2024). Consistency models for scalable and fast simulation-based inference. In *Advances in Neural Information Processing Systems*.

Sharrock, L., Simons, J., Liu, S., and Beaumont, M. (2024). Sequential neural score estimation: Likelihood-free inference with conditional score based diffusion models. In *Proceedings of the 41st International Conference on Machine Learning*.

Sisson, S. A., Fan, Y., and Beaumont, M. (2018). *Handbook of Approximate Bayesian Computation*. CRC Press.

Sutherland, D. J., Tung, H.-Y., Strathmann, H., De, S., Ramdas, A., Smola, A., and Gretton, A. (2017). Generative models and model criticism via optimized maximum mean discrepancy. In *International Conference on Learning Representations*.

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

Tejero-Cantero, A., Boelts, J., Deistler, M., Lueckmann, J.-M., Durkan, C., Gonçalves, P. J., Greenberg, D. S., and Macke, J. H. (2020). sbi: A toolkit for simulation-based inference. *Journal of Open Source Software*, 5(52):2505.

Theis, L., van den Oord, A., and Bethge, M. (2016). A note on the evaluation of generative models. In *International Conference on Learning Representations*.

Thomas, O., Dutta, R., Corander, J., Kaski, S., and Gutmann, M. U. (2022). Likelihood-free inference by ratio estimation. *Bayesian Analysis*, 17(1):1–31.

Vehtari, A., Gelman, A., Simpson, D., Carpenter, B., and Bürkner, P.-C. (2021). Rank-Normalization, Folding, and Localization: An Improved $\widehat{R}$ for Assessing Convergence of MCMC. *Bayesian Analysis*, 16(2):667 – 718.

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

Wildberger, J. B., Dax, M., Buchholz, S., Green, S. R., Macke, J. H., and Schölkopf, B. (2023). Flow matching for scalable simulation-based inference. In *Advances in Neural Information Processing Systems*.

Yao, Y. and Domke, J. (2023). Discriminative calibration: Check bayesian computation from simulations and flexible classifier. In *Advances in Neural Information Processing Systems*.

Yao, Y., Régaldo-Saint Blancard, B., and Domke, J. (2024). Simulation-based stacking. In *Proceedings of the 27th International Conference on Artificial Intelligence and Statistics*.

Zhao, S., Sinha, A., He, Y., Perreault, A., Song, J., and Ermon, S. (2022). Comparing distributions by measuring differences that affect decision making. In *International Conference on Learning Representations*.

Simulation-based Inference for High-dimensional Data using Surjective Sequential Neural Likelihood Estimation (Supplementary Material)

Simon Dirmeier[1,2] **Carlo Albert**[3] **Fernando Perez-Cruz**[2,4]

[1]Swiss Data Science Center, Switzerland
[2]ETH Zurich, Switzerland
[3]Swiss Federal Institute of Aquatic Science and Technology, Switzerland
[4]Bank for International Settlements, Switzerland

A BACKGROUND

A.1 NEURAL POSTERIOR ESTIMATION

Neural posterior estimation (SNPE) methods (Papamakarios and Murray, 2016; Lueckmann et al., 2017; Greenberg et al., 2019; Deistler et al., 2022; Wildberger et al., 2023) use a normalizing flow to directly target the posterior distribution thereby approximating $q_f(\theta|y) \approx p(\theta|y)$. SNPE-C (Greenberg et al., 2019)) uses the same sequential training procedure as SNL. In the first round, however, it optimizes the maximum likelihood objective $\mathbb{E}_{\mathcal{D}}\left[q_f(\theta|y)\right]$ in each round. Subsequent rounds proceed by first composing a proposal prior as $\hat{p}^r(\theta) = q_f(\theta|y_0)$, simulating new pairs $\{y_n, \theta_n\}^r_{1\ldots N}$ where $\theta_n \sim \hat{p}^r(\theta)$ and then re-training the NF. Since the parameters are sampled from the proposal prior $\hat{p}^r(\theta)$, the surrogate posterior would no longer target the true posterior $p(\theta|y)$ but rather

$$q_f(\theta|y) \propto p(\theta|y)\frac{\hat{p}^r(\theta)}{p(\theta)}$$

Greenberg et al. (2019) overcome this by deriving the new objective $\mathbb{E}_{\mathcal{D}}\left[\frac{1}{Z}q_f(\theta|y)\frac{\hat{p}^r(\theta)}{p(\theta)}\right]$ which however requires the computation of a normalization constant Z.

SNPE-C can simulate posterior realizations by sampling from the normalizing flow base distribution first, and then propagating the samples through the flow layers. This can lead to posteriors that are outside of the prior bounds which need to be rejected and the procedure repeated until a sample of desired size has been taken. Specifically, if the prior distributions are constrained, e.g., containing scale parameters, or are very narrow, APT is known to exert 'leakage', i.e., the posterior approximation might produce samples that are not within the prior bounds. In this case, the rejection rate of posterior samples is elevated, for instance, as reported in Durkan et al. (2020) or Glöckler et al. (2022), the latter of which having observed rejection rates of up to 99%, which necessitates the use of MCMC methods instead. Leakage significantly reduces the usefulness of SNPE methods in comparison to SNL where draws are generated using MCMC in the first place. Furthermore, for structured data sets, e.g., time series data, SNPE-C requires facilitating a second neural network to embed the data before conditioning which increases the number of effective parameters.

A.2 LIKELIHOOD RATIO ESTIMATION

Neural likelihood ratio estimation (NRE) methods (Hermans et al., 2020; Durkan et al., 2020; Miller et al., 2022; Delaunoy et al., 2022)) learn the likelihood-to-evidence ratio $r(y, \theta) = \frac{p(y|\theta)}{p(y)} = \frac{p(\theta|y)}{p(\theta)}$ and then build a surrogate posterior $\hat{p}(\theta|y) = \hat{r}(y, \theta)p(\theta)$. A major advantage of NRE methods is, that they do not need to train a model that estimates a density using normalizing flow which often brings significant computational and numerical advantages. While NRE-C (Miller et al., 2022) has been proposed in a non-sequential scenario, it is also possible to derive posterior distributions sequentially (SNRE-C; see, e.g., Tejero-Cantero et al. (2020)). In this case, since a proposal posterior $p^r(\theta|y_0)$ is derived after round r, which

changes the joint distribution to $p(y|\theta)p^r(\theta|y_0)$, the estimated ratio becomes

$$r(y,\theta) = \frac{p(y,\theta)}{p^r(\theta|y_0)} = \frac{p(\theta|y)}{p(\theta)}$$

Consequently, the true posterior can only be estimated up to a constant:

$$p(\theta|y) \propto r(y,\theta)p(\theta)$$

A.3 NOTES

The idea of using non-trivial embedding networks, such as CNNs or LSTMs for NPE and NRE methods is not new (see e.g., Greenberg et al. (2019) or the notebooks of the SBI Python package[1]). This requires an additional neural network and consequently increases the number of total parameters. We, on the other hand, do dimensionality reduction and likelihood estimation in one step and with one network.

B MATHEMATICAL DERIVATIONS

The derivation of the surjection layer used in SSNL largely follows the SurVAE framework of Nielsen et al. (2020) and Klein et al. (2021). The SurVAE framework models the log-probability $\log p(y)$ of a P-dimensional data point $y \in \mathcal{Y}$ as

$$\log p(y) = \log p\left(z\right) + V(y,z) + E(y,z), \qquad z \sim q(z|y) \tag{3}$$

where $q(z|y)$ is some amortized (variational) distribution, $z \in Z$ is a latent variable with distribution $p(z)$, $V(y,z)$ is denoted likelihood contribution term and $E(y,z)$ is a bound looseness term.

Nielsen et al. (2020) define the likelihood contribution for inference surjections as

$$V(y,z) = \lim_{q(z|y)\to\delta(z-h^{-1}(y))} \mathbb{E}_{q(z|y)}\left[\log \frac{p(y|z)}{q(z|y)}\right]$$

where $p(y|z)$ is some generative stochastic transformation, $h^{-1} : \mathcal{Y} \to \mathcal{Z}$ is an inference surjection and where we for convenience of notation denote with $h : \mathcal{Z} \to \mathcal{Y}$ a right inverse function to h^{-1}. For bijective normalizing flows the bound looseness term equals $E(y,z) = 0$. For surjective normalizing flows, the same is true if a right inverse h exists (i.e., when the stochastic right inverse condition is satisfied).

By observing (see also Appendix A of Nielsen et al. (2020) and main manuscript Klein et al. (2021)) that the composition of a differentiable function g with a Dirac δ function and a bijection f is

$$\int \delta\left(g\left(y\right)\right) f\left(g\left(y\right)\right) \left|\det \frac{\partial g\left(y\right)}{\partial y}\right| \mathrm{d}y = \int \delta\left(u\right) f\left(u\right) \mathrm{d}u$$

we can conclude that

$$\delta\left(g\left(y\right)\right) = \delta\left(y - y_0\right) \left|\det \frac{\partial g\left(y\right)}{\partial y}\right|^{-1}_{y=y_0}$$

where y_0 is the root of g (which assumes that f has compact support, the root is unique and that the Jacobian is not singular).

We now define a conditional bijection $f(z; y_-, \theta)$ and its inverse $f^{-1}(y_+; y_-, \theta)$ for any $Q < P$, set $g(y) = z - f^{-1}(y_+; y_-, \theta)$ (which has its root at $y_0 = f(z; y_-, \theta)$) and define

$$\begin{aligned} q(z|y) &= \delta\left(z - f^{-1}(y_+; y_-, \theta)\right) \\ &= \delta\left(y_+ - f(z; y_-, \theta)\right) \left|\det J^{-1}\right|^{-1} \end{aligned}$$

[1] https://sbi-dev.github.io/sbi/tutorial/05_embedding_net/

where

$$J^{-1} = \left.\frac{\partial f^{-1}(y_+; y_-, \theta)}{\partial y_+}\right|_{y_+ = f(z; y_-, \theta)}$$

Using this result and the conditional distribution $p(y|z) = p(y_-|z, \theta)$ the likelihood contribution for a surjection layer becomes

$$\begin{aligned}
V(y, z) &= \lim_{q(z|y) \to \delta(z - h^{-1}(y))} \mathbb{E}_{q(z|y)} \left[\log \frac{p(y|z)}{q(z|y)} \right] \\
&= \int \delta\left(z - f^{-1}(y_+; y_-, \theta)\right) \log \frac{p(y_-|z, \theta)}{\delta\left(z - f^{-1}(y_+; y_-, \theta)\right)} \mathrm{d}z \\
&= \int \delta\left(y^+ - f(z; y_-, \theta)\right) |\det J^{-1}|^{-1} \log \frac{p(y^-|z, \theta)}{\delta(y^+ - f(z; y_-, \theta))|\det J^{-1})|^{-1}} \mathrm{d}z \\
&= \int \delta\left(y^+ - \tilde{y}^+\right) \log \frac{p(y^-|z, \theta)}{\delta(y^T - \tilde{y}^+)|\det J^{-1}|^{-1}} \mathrm{d}\tilde{y}^+ \\
&= \log p\left(y_-|f^{-1}(y_+; y_-, \theta)\right) - \log \left|\det J^{-1}\right|^{-1}
\end{aligned}$$

where we used the change of variables $\tilde{y}^+ = f(z; y_-, \theta)$ yielding $\mathrm{d}\tilde{y}^+ = \mathrm{d}z|\det J^{-1}|^{-1}$.

C IMPLEMENTATION DETAILS

Surjection layers can be implemented in a straight-forward manner by extending the bijection layers of conventional machine libraries. Below, we demonstrate the implementation of a conditional affine masked autoregressive surjective flow that uses an affine MAF (Papamakarios et al., 2017), called `AffineMaskedAutoregressive` as a super class.

```python
@dataclass
class AffineMaskedAutoregressiveSurjection(AffineMaskedAutoregressive):
    n_keep: int
    decoder: Callable
    conditioner: MADE

    def _inner_bijector(self):
        # define the bijector 'f'
        return AffineMaskedAutoregressive(self.conditioner)

    def _inverse_and_likelihood_contribution(self, y, x=None, **kwargs):
        # here, we define the subsets by just splitting y after some index
        # in general, we do it as describe it as in the main manuscript
        y_plus, y_minus = y[..., :self.n_keep], y[..., self.n_keep:]
        y_cond = y_minus

        if x is not None:
            y_cond = jnp.concatenate([y_cond, x], axis=-1)
        # compute lower-dimensional representation
        z, jac_det = self._inner_bijector().inverse_and_log_det(y_plus, y_cond)

        z_condition = z
        if x is not None:
            z_condition = jnp.concatenate([z, x], axis=-1)
        # compute conditional probability
        lc = self.decoder(z_condition).log_prob(y_minus)

        return z, lc + jac_det
```

where `MADE` is a masked autoencoder for density estimation (Germain et al., 2015), `decoder` corresponds to the conditional density $p(y_-|z, \theta)$.

D EXPERIMENTAL DETAILS

D.1 IMPLEMENTATION DETAILS

All models are implemented using the Python packages `sbijax`, (Dirmeier et al., 2024), `surjectors` (Dirmeier, 2024), the SBI toolbox (Tejero-Cantero et al., 2020), and the Deepmind JAX ecosystem (Bradbury et al., 2018; Babuschkin et al., 2020). We simulate data from stochastic differential equations using the package Diffrax (Kidger, 2021).

D.2 TRAINING AND SAMPLING

We trained each model using an Adam optimizer with fixed learning rate of $r = 0.0001$ and momentums $b_1 = 0.9$ and $b_2 = 0.999$. Each experiment uses a mini-batch size of 100. The optimizer is run until a maximum of 2000 epochs is reached or no improvement on a validation set can be observed for 10 consecutive iterations. The validation set consists of 10% of the entire data set, while the other 90% are used for training. For each round, we start training the neural network from scratch and do not continue from the previously learned state. Each model was trained on a HPC computing cluster using a single node consisting of two 18 core Broadwell CPUs (Intel Xeon E5-2695 v4).

We train each method in $R = 15$ rounds. Each round a new set of pairs $\{(y_n, \theta_n)\}_{n-1}^N$ of size N is generated using draws from the prior $\theta_n \sim p(\theta)$ and simulator $y_n \leftarrow sim(\theta_n)$, and then used for training the density estimators or classifier, respectively.

For SSNL and SNL, we used the No-U-turn sampler (Hoffman and Gelman, 2014) from the sampling library BlackJAX to sample from the intermediate and final posterior distributions $p^r(\theta|y_0)$ using 4 chains of a fixed length of $10\,000$ each of which the first 5000 iterations are discarded as burn-in per chain. For SNPE-C and SNRE-C experiments, we use the slice sampler of the SBI toolbox for sampling (which we do in lieu of rejection sampling to avoid leakage; see the Appendix A).Samples from the "true" posterior distribution have been drawn using TensorFlow Probability's slice sampler where we used 10 chains of length $20\,000$ of which we discarded the first $10\,000$ as burn-in. Convergence of the true posteriors in this case has been diagnosed using the potential scale reduction factor (Gelman and Rubin, 1992; Vehtari et al., 2021), effective sample size calculations, and conventional graphical diagnostics (Gabry et al., 2019).

D.3 NEURAL NETWORK ARCHITECTURES

For all experiments and evaluated methods, we used the same neural network architectures. We followed the neural network architectures as described in Greenberg et al. (2019); Papamakarios et al. (2019); Miller et al. (2022) and tried to keep the number of total parameters of each model as comparable as possible to allow for an unbiased evaluation.

SSNL The SSNL architectures use a total of $K = 5$ layers, the third of which is a surjection layer with reduction factors of 25%, 50% or 75% which we chose arbitrarily. For instance, for a reduction factor of 25%, we take the initial dimensionality P and reduce it to $Q = \lfloor 0.25 * P \rfloor$. Each of the layers is parameterized by a MAF which uses a MADE network with two layers with 50 neurons each as conditioner (Germain et al., 2015; Papamakarios et al., 2017). The MAFs use tanh activation functions. The conditional densities are parameterized using a two-layer MLP with tanh activation functions. Between each MAF layer, we add a permutation layer that reverses the vector dimensions. In practice, we assume that optimizing the number of surjection layers and their reduction factors is advisable. This can, for instance, be done empirically by examining the likelihood profiles during training or by reducing to the same order of magnitude as required summary statistics.

SNL SNL uses a total of $K = 5$ layers. Each of the layers is parameterized by a MAF which uses a MADE network with two layers with 50 neurons each as conditioner (Germain et al., 2015; Papamakarios et al., 2017) with permutations in between which reverse the vector dimensions. SNL uses tanh activation functions.

SNPE-C SNPE-C uses the same normalizing flow architecture as SNL. We, otherwise, use the default SNPE-C parameterisation of the SBI toolbox which uses 10 atoms for classification.

SNRE-C SNRE-C architectures consist of MLP networks with two layers and 50 nodes per layers. SNRE-C uses ReLU activation functions. We, otherwise, use the default SNRE-C parameterisation of the SBI toolbox which uses 5 classes to classify against and $\gamma = 1.0$.

SNASS To keep the number of parameters as equal as possible, SNASS uses a normalizing flow using three flow layers each consisting of a MADE with two layers and 50 nodes each. SNASS uses as summary and critic networks two MLPs with two hidden layers and 50 nodes each. All activation functions are tanhs.

SNASSS Similarly, SNASS uses a normalizing flow using two flow layers each consisting of a MADE with two layers and 50 nodes each. SNASSS uses as summary and critic networks three MLPs with a single hidden layer and 50 nodes. All activation functions are tanhs.

D.4 ESTIMATION OF DIVERGENCES

Zhao et al. (2022) propose using the H-divergence

$$D_\ell^\phi(p||q) = \phi\left(H_\ell\left(\frac{p+q}{2}\right) - H_\ell(p), H_\ell\left(\frac{p+q}{2}\right) - H_\ell(q)\right)$$

to compare two empirical distributions. Here, $H_\ell(p) = \inf_{a\in\mathcal{A}}\mathbb{E}_p[\ell(X,a)]$ is the Bayes optimal loss of some decision function over an action set $\mathcal{A}$.

They further illustrate the H-Min divergence

$$D_l^{\text{Min}} = H_\ell\left(\frac{p+q}{2}\right) - \min\left(H_\ell(p), H_\ell(q)\right)$$

and H-Jensen Shannon divergence

$$D_\ell^{\text{JS}} = H_\ell\left(\frac{p+q}{2}\right) - \frac{1}{2}\left(H_\ell(p), H_\ell(q)\right)$$

as two special cases.

We compute H_ℓ using the negative log-likelihood of kernel density estimators (KDEs) using 5-fold cross-validation. Specifically, we first fit KDE with Gaussian kernels on samples of size $N = 10\,000$ from the true posterior distribution p $\mathcal{P} = \{\theta_n^{\text{true}}\}_{n=1}^N$ and surrogate posterior distribution q $\mathcal{Q} = \{\theta_n^{\text{surrogate}}\}_{n=1}^N$ separately (i.e., one KDE for each set of samples). We find the optimal hyperparameters $\phi \in \{0.1, \ldots, 5\}$ for each KDE using a grid search. We then, in $F = 5$ different iterations (folds), subsample $\mathcal{S}_f \subset \mathcal{S}$ and $\mathcal{T}_f \subset \mathcal{T}$ randomly (i.e., with equal probability of being in the subsample) and estimate a KDE on $\mathcal{F}_f = \mathcal{S}_f \cup \mathcal{T}_f$. We compute estimates of H_l via

$$\begin{aligned} H_\ell(p) &\approx \inf_a \frac{1}{N}\ell(\mathcal{P}, a) \\ H_\ell(q) &\approx \inf_a \frac{1}{N}\ell(\mathcal{Q}, a) \\ H_\ell^f\left(\frac{p+q}{2}\right) &\approx \inf_a \frac{1}{N}\ell(\mathcal{S}_f, a) \end{aligned}$$

where $\ell(\mathcal{X}, a)$ is the negative log-likelihood of the data set $\mathcal{X}$ given the optimized hyperparameters (action) a.

D.5 ESTIMATION OF INTRINSIC DIMENSIONALITY

We computed the intrinsic dimensionality of a data set using the "tight local intrinsic dimensionality estimator" (TLE) algorithm (Amsaleg et al., 2022). The TLE is an estimator of the local intrinsic dimensionality, i.e., the intrinsic dimension

of each data point in a data set. Mathematically, the local intrinsic dimensionality for a data point x w.r.t to the distance $r := r(x)$ to its k nearest neighbors is defined as

$$\mathrm{ID}(x) = \lim_{r \to 0} \lim_{\epsilon \to 0} \frac{\log(F((1+\epsilon) \cdot r)/F(r))}{\log(1+\epsilon)}$$

where we denote with $F(r)$ the cdf of R which can be estimated empirically. The intrinsic dimension of a data point x describes the relative rate at which $F(r)$ increases.

For details and the derivation of the TEL intrinsic dimensionality estimator, we refer the reader to Amsaleg et al. (2022).

For Figure 5c, we simulated $N = 1000$ observations $\{y_n\}_{n=1}^N$ from the generative models of the Ornstein-Uhlenbeck, Lotka-Volterra and SIR models, respectively, and estimated the local intrinsic dimensions of data set using the Python package `scikit-dimension` (Bac et al., 2021). The package contains several different estimators for local intrinsic dimensionality, and we chose the TLE estimator arbitrarily.

D.6 SOURCE CODE

Source code including detailed instructions to reproduce and replicate all experiments can be found in the supplemental material or on GitHub at `github.com/dirmeier/ssnl`.

E ADDITIONAL DETAILS ON EXPERIMENTAL MODELS

This section describes the nine experimental models in more detail.

E.1 SIMPLE LIKELIHOOD COMPLEX POSTERIOR

The simple likelihood complex posterior (SLCP, Papamakarios et al. (2019)) model with 8 dimensions uses the following generative process

$$\begin{aligned}
\theta_i &\sim \text{Uniform}(-3, 3) \text{ for } i = 1, \dots, 5 \\
\mu(\theta) &= (\theta_1, \theta_2), \phi_1 = \theta_3^2, \phi_2 = \theta_4^2 \\
\Sigma(\theta) &= \begin{pmatrix} \phi_1^2 & \tanh(\theta_5)\phi_1\phi_2 \\ \tanh(\theta_5)\phi_1\phi_2 & \phi_2^2 \end{pmatrix} \\
y_j|\theta &\sim \mathcal{N}(y_j; \mu(\theta), \Sigma(\theta)) \text{ for } j = 1, \dots, 4 \\
y &= [y_1, \dots, y_4]^T
\end{aligned}$$

The SLCP generally favours neural likelihood methods of neural posterior methods, since modelling a simple likelihood and then sampling from a multi-model posterior is easier in comparison to vice versa. For each round r, we generated 1000 pairs $\{(y_n, \theta_n)\}$ from the SLCP model.

E.2 ORNSTEIN-UHLENBECK

The Ornstein-Uhlenbeck (OU) process (Särkkä and Solin, 2019) is a one-dimensional stochastic differential equation that models velocity of a particle suspended in a medium. It has the following form:

$$dY_t = -\theta_2(Y_t - \theta_1)\mathrm{d}t + \theta_3 \mathrm{d}W_t$$

where W_t is a Wiener process and θ are the parameters of interest. The presentation of the OR process above uses an additional drift term θ_2.

Instead of solving this SDE numerically, the OR process admits the analytical forms

$$Y_t \mid Y_0 = y_0 \sim \mathcal{N}\left(\theta_1 + (y_0 - \theta_1)e^{-\theta_2 t}, \tfrac{\theta_3^2}{2\theta_2}(1 - e^{-2\theta_2 t})\right)$$

and

$$Y_t \mid Y_s = y_s \sim \mathcal{N}\left(\theta_1 + (y_0 - \theta_1)e^{-\theta_2(t-s)}, \frac{\theta_3^2}{2\theta_2}(1 - e^{-2\theta_2(t-s)})\right)$$

where $s < t$. The conditional density above can be used both for sampling and evaluating the density of an observation Y_t. We simulate the OU process for each experiment using the generative model

$$\begin{aligned}
\theta_1 &\sim \mathcal{U}(0, 10) \\
\theta_2 &\sim \mathcal{U}(0, 5) \\
\theta_3 &\sim \mathcal{U}(0, 2) \\
Y_t \mid Y_s = y_s &\sim \mathcal{N}\left(\theta_1 + (y_0 - \theta_1)e^{-\theta_2(t-s)}, \frac{\theta_3^2}{2\theta_2}(1 - e^{-2\theta_2(t-s)})\right)
\end{aligned}$$

and initialize $y_0 = 0$. We sample 100 equally-spaced observations Y_t where $t \in \{0, \ldots, 10\}$. The conditional density can be used both for sampling and evaluating the density of an observation Y_t.

E.3 LOTKA-VOLTERRA

The Lotka-Volterra model is a model from ecology that describes the dynamics of a "prey" population and a "predator" population:

$$\begin{aligned}
\theta_1 &\sim \text{LogNormal}(-0.125, 0.5) \\
\theta_2 &\sim \text{LogNormal}(-3, 0.5) \\
\theta_3 &\sim \text{LogNormal}(-0.125, 0.5) \\
\theta_4 &\sim \text{LogNormal}(-3, 0.5) \\
\tfrac{\mathrm{d}X_1}{\mathrm{d}t} &= \theta_1 X_1 - \theta_2 X_1 X_2 \\
\tfrac{\mathrm{d}X_2}{\mathrm{d}t} &= -\theta_3 X_1 + \theta_4 X_1 X_2 \\
(Y_{t1}, Y_{t2}) &\sim \text{LogNormal}\left(\log\left(X_{t1}, X_{t2}\right), 0.1\right)
\end{aligned}$$

where X_1 are is the density of the prey population and X_2 is the density of some predator population. The parameter $\theta = [\theta_1, \ldots, \theta_4]^T$ describes growth and death rates, respectively, and effects of presence of predators and prey, respectively.

We follow the parameterization in Lueckmann et al. (2021), but sample a longer time series, i.e., 50 equally-spaced observations Y_t where $t \in [0, 30]$. We then concatenate the two 50-dimensional vectors $y_t = [y_{t1}^T, y_{t2}^T]^T$ yielding a 100-dimensional observation.

We solve the ODE using the Python package Diffrax using a Tsit5 solver.

E.4 SIR MODEL

The SIR model is a model from epidemiology that describes the dynamics of the number of individuals in three compartmental states (susceptible, infectious, or recovered) which is, for instance, be aplpied to model the spread of diseases. We again adopt the presentation by Lueckmann et al. (2021) which defines the generative model

$$\begin{aligned}
\theta_1 &\sim \text{LogNormal}(\log(0.4), 0.5) \\
\theta_2 &\sim \text{LogNormal}(\log(1/8), 0.2) \\
\frac{\mathrm{d}S}{\mathrm{d}t} &= -\theta_1 \frac{SI}{N} \\
\frac{\mathrm{d}I}{\mathrm{d}t} &= \theta_1 \frac{SI}{N} - \theta_2 I \\
\frac{\mathrm{d}R}{\mathrm{d}t} &= \theta_2 I \\
Y_t &\sim \text{Binomial}\left(1000, \frac{I_t}{N}\right)
\end{aligned}$$

where we set $N = 1\,000\,000$, and the initial conditions $s_0 = N - 1$, $i_0 = 1$ and $r_0 = 0$. We sample 100 evenly-spaced observations Y_0 where $t \in \{0, \ldots, 160\}$.

Previous work, e.g., Lueckmann et al. (2021), used continuous normalizing flows (i.e., pushforwards from a continuous base distribution, not continuous normalizing flows as in Chen et al. (2018); Grathwohl et al. (2019)). Continuous likelihood-based models, such as SNL using MAFs, cannot adequately represent discrete data. As a remedy, we dequantize the counts uniformly after sampling them (Theis et al., 2016), i.e., we add noise $u_t \sim \mathcal{U}(0,1)$, such that $\tilde{y}_t = y_t + u_t$ and use the noised data for trained. While other approaches to dequantization, such as Ho et al. (2019), would possibly be more rigorous, we found that this simple approach suffices.

We solve the ODE using the Python package Diffrax using a Tsit5 solver.

E.5 BETA GENERALIZED LINEAR MODEL

We evaluated SSNL and the three baselines against a Beta generalized linear regression model (Beta GLM). We use the following generative model

$$\begin{aligned} &\theta \sim \mathcal{N}(0, B) \\ &\eta = X\theta, \qquad \mu = \text{sigmoid}(\eta) \\ &Y \sim \text{Beta}(\mu c, (1-\mu)c) \end{aligned}$$

where $c \in \mathbb{R}^+$ is a non-negative concentration parameter, B is computed as in Appendix T.6 of Lueckmann et al. (2021), Beta describes a Beta distribution with a mean-concentration parameterization (Ferrari and Cribari-Neto, 2004)

In Lueckmann et al. (2017), a Bernoulli GLM is used instead of a Beta GLM(called Bernoulli GLM raw). We changed the Bernoulli likelihood to a Beta likelihood because of the the same reason as for the Lotka-Volterra model (a CNF can generally not model a discrete distribution). We followed the implemtation in SBIBM (`https://github.com/sbi-benchmark/sbibm`) as a design matrix B.

E.6 GAUSSIAN MIXTURE MODEL

The Gaussian mixture (GGM) described in "Negative examples and limitations" in Section 4 uses the following generative process:

$$\begin{aligned} \theta &\sim \mathcal{U}(-10, 10) \\ Y \mid \theta &\sim 0.5\mathcal{N}(\theta, I) + 0.5\mathcal{N}(\theta, \sigma^2 I) \end{aligned}$$

where $\sigma^2 = 0.01\ I$ is a unit matrix, and both $\theta \in \mathbb{R}^2$ and $Y \in \mathbb{R}^2$ are two-dimensional random variables. The GGM again follows the representation in Lueckmann et al. (2021).

E.7 HYPERBOLOID

The hyperboloid model (Forbes et al., 2022) described in "Negative examples and limitations" in Section 4 is a 2-component mixture model of t-distributions of the form

$$\begin{aligned} \theta &\sim \mathcal{U}(-2, 2) \\ Y \mid \theta &\sim \frac{1}{2}t(\nu, F(\theta; a)\mathbb{I}, \sigma^2 I) + \frac{1}{2}t(\nu, F(\theta; b)\mathbb{I}, \sigma^2 I) \end{aligned}$$

where t represents a Student's t-distribution with ν degrees of freedom, mean $F(\theta; x) = (||\theta - x_1||_2 - ||\theta - x_2||_2)$ and scale matrix $\sigma^2 I$ and $\mathbb{I}$ is vector of ones. We follow Forbes et al. (2022), and in our experiments set $\theta \in \mathbb{R}^2$ to be uniformly distribution, $a_1 = [-0.5, 0.0]^T$, $a_2 = [0.5, 0.0]^T$, $b_1 = [0.0, -0.5]^T$, $a_2 = [0.0, 0.5]^T$, $\nu = 3$ and $\sigma^2 = 0.01$.

E.8 SOLAR DYNAMO

The solar dynamo model is a non-linear time series model with both additive and multiplicative noise terms

$$\begin{aligned}
\theta_1 &\sim \mathcal{U}(0.9, 1.4) \\
\theta_2 &\sim \mathcal{U}(0.05, 0.25) \\
\theta_3 &\sim \mathcal{U}(0.02, 0.15) \\
g(y) &= \frac{1}{2}[1 + \text{erf}(\tfrac{y-b_1}{w_1})][1 - \text{erf}(\tfrac{y-b_2}{w_2})] \\
\alpha_i &\sim \mathcal{U}(\theta_1, \theta_1 + \theta_2) \\
\epsilon_i &\sim \mathcal{U}(0, \theta_3) \\
y_{t+1} &\leftarrow \alpha_i g(y_t) y_t + \epsilon_i
\end{aligned}$$

where $\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x \exp(-t^2)\mathrm{d}y$ is the Gauss error function. We simulate a time series of length $N = 100$ recursively starting from $y_0 = [1, 1]^T$. We follow Albert et al. (2022), and set $b_1 = 0.6$, $w_1 = 0.2$, $b_2 = 1$ and $w_2 = 0.8$.

E.9 NEURAL MASS MODEL

The stochastic version of the Jansen-Rit neural mass model (Ableidinger et al., 2017) describes the collective electrical activity of neurons. The model is a 6-dimensional stochastic differential equation of the form

$$\begin{aligned}
\theta_1 &\sim \mathcal{U}(10, 250) \\
\theta_2 &\sim \mathcal{U}(50, 500) \\
\theta_3 &\sim \mathcal{U}(100, 5000) \\
\theta_4 &\sim \mathcal{U}(-20, 20) \\
\mathrm{d}\begin{pmatrix} Q_t \\ P_t \end{pmatrix} &= \begin{pmatrix} P_t \\ -\Gamma^2 Q_t - 2\Gamma P_t + G_\theta(Q_t, \theta) \end{pmatrix} \mathrm{d}t + \begin{pmatrix} 0 \\ \Sigma_\theta \end{pmatrix} \mathrm{d}W_t
\end{aligned}$$

The actual signal $Y = 10^{g/10}(X_{t1} - X_{t2})$ where $Q = [Y_1, Y_2, Y_3]^T$, $P = [Y_4, Y_5, Y_6]^T$ and W_t is a Wiener process.$\Sigma_\theta = \text{diag}(\sigma_4, \sigma_5, \sigma_6)$ and $\Gamma = \text{diag}(a, a, b)$ are diagonal 3×3 matrices with positive a and b. The vector

$$G(Q_t; \theta) = \begin{pmatrix} Aa[\text{sig}(X_2 - X_3)] \\ Aa[\mu + C_2 \text{sig}(C_1 X_1)] \\ Bb[C_4 \text{sig}(C_3 X_1)] \end{pmatrix}$$

is a 3-dimensional vector of displacement terms and

$$\text{sig}(x) = \frac{v_{\text{max}}}{1 + \exp(r(v_0 - x))}$$

We are interest in inference of the 4-dimensional vector $\theta = [\theta_1, \ldots, \theta_4]^T = [C, \mu, \sigma, g]^T$. The parameters C_i are related via $C_1 = \theta_1$, $C_2 = 0.8\theta_1$, $C_3 = 0.25$ and $C_4 = 0.25\theta_1$. The other parameters are $\mu_4 = \theta_2$, $\sigma_5 = \theta_3$ and $g = \theta_4$. Following previous work, we initialize $y_0 = [0.08, 18, 15, -0.5, 0, 0]^T$ and simulate a time series Y_t with $t \in [0, 8]$ with sampling frequency $Hz = 512$. We then takes 100 equally-spaced elements from Y_t.

We refer the reader to Ableidinger et al. (2017), Rodrigues et al. (2021) and Buckwar et al. (2020) for detailed explanations of all constants and equations from where we also adopted the parameterization: $A = 3.25$, $B = 22$, $a = 100$, $b = 50$, $v_{\text{max}} = 5$, $v_0 = 6$, $r = 0.56$, $\sigma_4 = 0.01$ and $\sigma_6 = 1$ (see also Linhart et al. (2023)).

We solve the SDE with the Python library `jrnmm` using the Strang-splitting method as described in Buckwar et al. (2020).

F ADDITIONAL RESULTS

This section presents additional results for the benchmark models.

F.1 FOUR SBI MODELS BENCHMARK

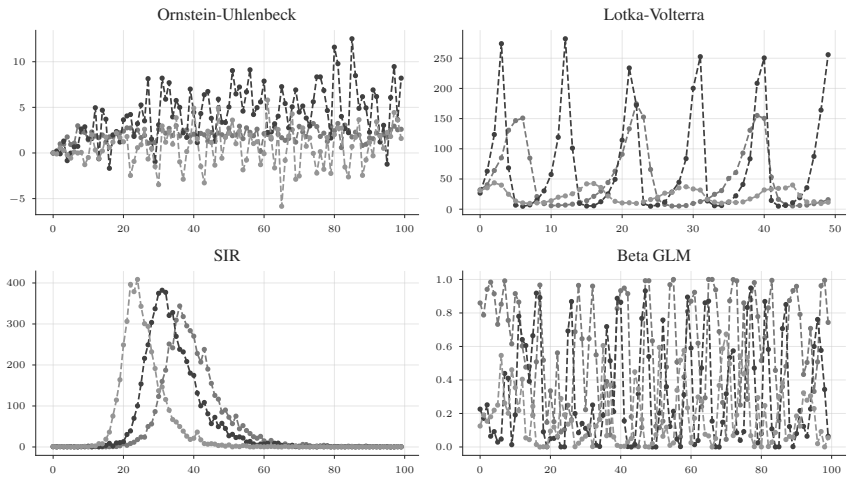


Figure 7: Data visualisations of the four benchmark models, Ornstein-Uhlenbeck, Lotka-Volterra, SIR and Beta GLM. For the first three models, the x-axis corresponds to the time index of the time series. For the Beta GLM, the x-axis only serves as an index.

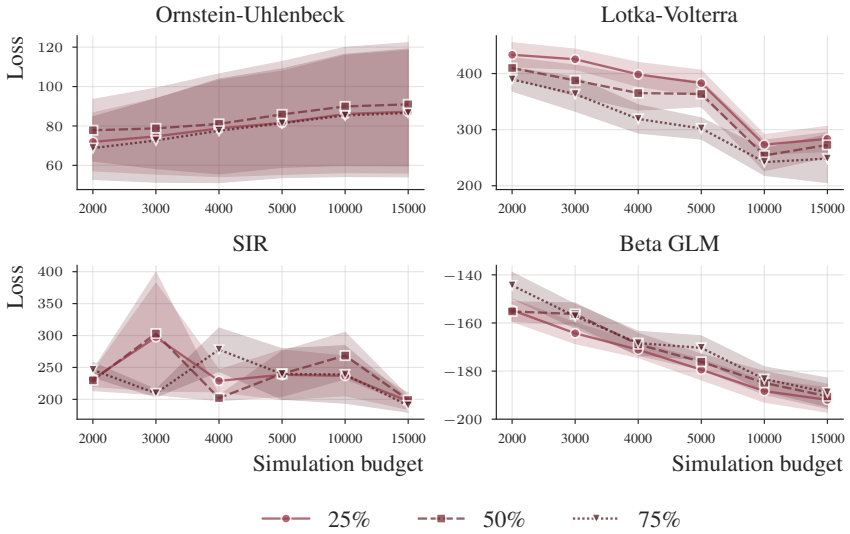
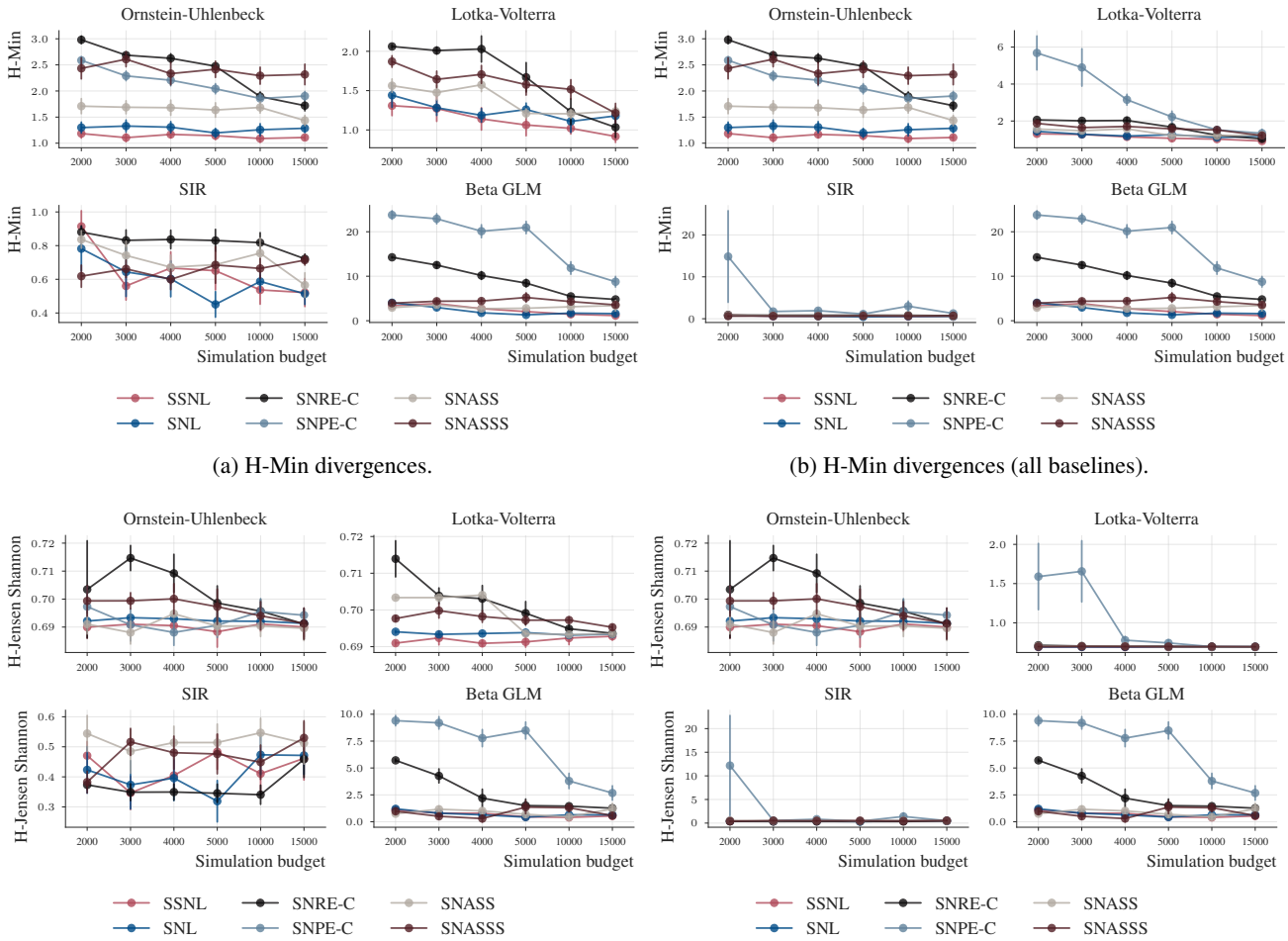


Figure 8: Likelihood profiles for the four benchmark models, Ornstein-Uhlenbeck, Lotka-Volterra, SIR and Beta GLM with different reduction factors. Each profile corresponds to the likelihood on the validation set used during training. Using the validation set is a fairly ad-hoc approach and could be done more rigorously, e.g., by using an additional test set where the loss is evaluated instead. Since the data is generated iid, however, this would not arguably not change much. The profiles for these datasets are very similar, only the LV model benefits significantly from different surjection layer dimensionalities.



(a) H-Min divergences.

(b) H-Min divergences (all baselines).

(c) H-Jensen Shannon divergences.

(d) H-Jensen Shannon divergences (all baselines).

Figure 9: H-Min and H-Jenson Shannon divergences of the Ornstein-Uhlenbeck, Lotka-Volterra, SIR and Beta GLM models (left withouth SNPE-C for Lotka-Volterra and SIR, right with all baselines). SSNL consistently outperforms all five baselines on Ornstein-Uhlenbeck, Lotka-Volterra, is on par with SNL on Beta GLM and displays mixed results on SIR on both divergence measure. Given that SSNL requires less parameters than SNL, SSNL has the clear advantage in Ornstein-Uhlenbeck, Lotka-Volterra and Beta GLM. The two divergences are consistent for three of the four models, for SIR the H-Min and H-Jensen Shannon show inconsistent divergences.

F.2 NEGATIVE EXAMPLES

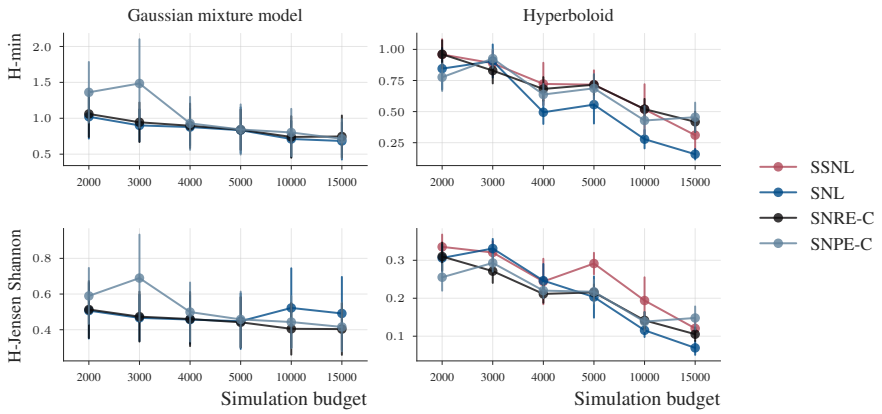


Figure 10: Negative examples. We show the H-Min and H-Jensen Shannon divergences on the Gaussian mixture and hyperboloid models. In both cases, SSNL can not outperform the three baselines. Since all data dimensions are informative of posterior parameters, reducing the dimensionality is theoretically only detrimental to the inferences. We did not conduct experiments on SNASS and SNASSS here, since their is no dimensionality reduction necessary.

F.3 SOLAR DYNAMO MODEL

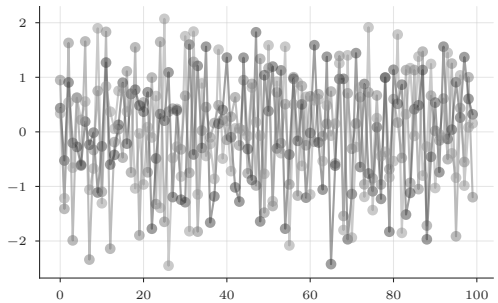


Figure 11: Data visualisations of the solar dynamo models. The x-axis represents index of the time series t, the y-axis the observed time point y_t.

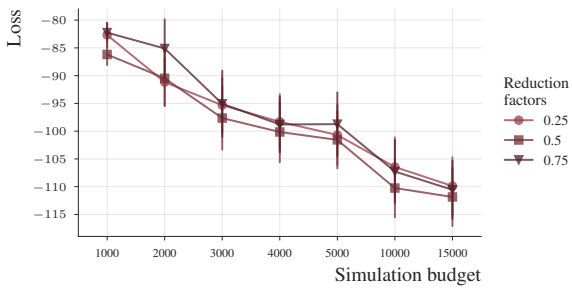


Figure 12: Likelihood profiles for the solar dynamo model with different reduction factors. Each profile corresponds to the likelihood on the validation set used during training. The likelihood profiles all show very similar losses (see y-axis). As a consequence, we used the model with the greatest reduction on dimensionality, i.e., 25% which reduces the dimensionality in the embedding layer to $Q = 25$.

F.4 JANSEN-RIT NEURAL MASS MODEL

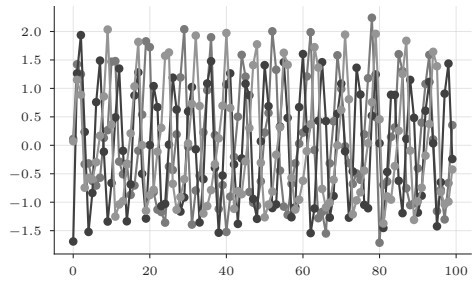


Figure 13: Data visualisations of the solar dynamo models. The x-axis represents index of the time series t, the y-axis the observed time point y_t.

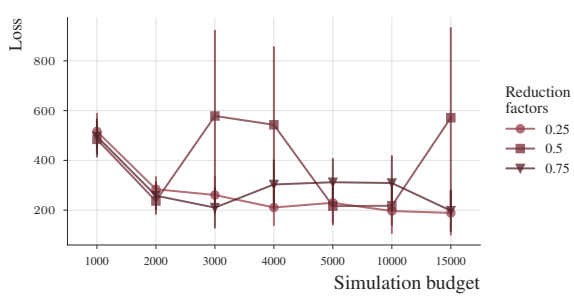


Figure 14: Likelihood profiles for the Jansen-Rit model with different reduction factors. Each profile corresponds to the likelihood on the validation set used during training. We used the model with the greatest reduction on dimensionality, i.e., 25%, which reduces the dimensionality in the embedding layer to $Q = 25$.