Biologically-inspired Deep-learning Workshop

Laura Bernáez Timón[1], Maximilian F. Eggl[2], Pedro Gonçalves[3]
Anastasia Krouglova[3], Guy Moss[4]

Abstract

A specialized workshop, the second in a series of workshops, for young researchers took place in the spring of 2024 in Guntersblum. This was funded by the "Begegnungszonen" program of the Joachim Herz Stiftung which aims to "support the design and implementation of interdisciplinary events for young scientists" and perfectly aligns with the intended goals of the workshop. By providing 16 doctoral students and postdoctoral researchers the opportunity to be exposed to state-of-the-art methods at the cutting edge of machine learning and life sciences, an interdisciplinary environment was created that allowed them to learn things beyond the fields they normally work in. The speaker for the 2024 workshop on the topic of the Simulation-based inference was Prof. Pedro Gonçalves of the Neuro-Electronics Research Flanders (NERF) Institute. He was assisted by his doctoral student, Anastasia Krouglova, and doctoral student, Guy Moss, from the lab of Prof. Jakob Macke (University of Tübingen). This report briefly introduces the workshop series' underlying idea, a motivation for the study and the importance and impact of Simulation-based inference, and a representative exposé of the students' work during the workshop.

1 The Workshop format

The exchange with established researchers and early-career scientists is invaluable to scientific progress. It allows new ideas to circulate through the community while providing a network for new and upcoming talent. To foster this exchange and enable interdisciplinary networking among early career scientists and renowned experts, a workshop was held in Guntersblum, Germany, and funded by the "Begegnungszonen" program of the Joachim Herz Foundation.

This second edition of the "Bio-inspired deep learning" series features a format that is not common in the neuroscientific community and was chosen to enhance the exchange between scientists at different career stages while providing an avenue for participants to learn about state-of-the-art research at the intersection of the life sciences and machine learning. This format, which was successfully implemented in the first iteration of workshop series, is based on a short timeframe (3 days) and encourages direct learning akin to on-the-job training. Hands-on demonstration by working through small-scale yet meaningful examples leads to pedagogical exposure to advanced concepts that may be difficult to understand if solely covered in lectures. By divided the workshop participants into small groups according to their skill level, experience and scientific maturity, a collegial environment is formed that encourages exchange and peer-based learning.

Additionally, the format is designed to be fundamentally interdisciplinary. Interdisciplinarity is becoming increasingly important in an interconnected science world, and thus, by bringing together scientists from a variety of different fields, the workshop provides a robust exchange of ideas and technologies from different scientific disciplines that enables the participants to become better scientists.

In contrast, the current standard format of workshops in neuroscience consists of longer time formats, several invited speakers to guide participants and a more lecture-based approach to learning.

[1]Mimotype Tech
[2]Instituto de Neurociencias, CSIC-UMH
[3]NERF
[4]University of Tübingen

This format has proven effective, but the format's passive learning and larger number of experts means that direct interactions between students and established researchers may be lost. Therefore, our workshop series aims to fill a gap in the currently available neuroAI options and provide a different avenue for early-stage scientists to network and learn new techniques.

The format of this workshop is directly inspired by the Young ERCOFTAC workshop series, an initiative of Peter Schmid, Shervin Bagheri and Taraneh Sayadi, and which M. Eggl had the pleasure of attending during his PhD. This ongoing workshop series provides a platform for early-career researchers to be exposed to state-of-the-art research within the fluid dynamics community.

We mirrored their approach by inviting 15-20 PhD students and Postdoctoral Fellows to the Weingut Domhof, a small remote location 30 km south of Mainz. The Domhof, a historical winery, has been converted into a hotel while still retaining its wine-making tradition. It features a quiet and rural surrounding, ideal for work in a secluded yet pleasant environment. The remote nature of the accommodation requires every student's active participation in daily activities, such as communal meals. These activities further contribute to the quick and lasting bonding of the participants that often lasts beyond the duration of the workshop.

The scientific program format was then devised to introduce the students (who had very different backgrounds and were at various stages of their scientific careers) to the workshop topic and prepare them for the group work in the days ahead. After an initial set of introductory talks on machine learning (M. Eggl) and Simulation-based inference (SBI) (P. Gonçalves), the organizing committee (L. Bernaez Timon and M. Eggl) and P. Gonçalves presented each group with their research topic, which covered a sub-topic of SBI. The projects were designed to require differing expertise and their scope was confined to tasks that could be feasibly accomplished solely using laptops. While working in their groups, students were closely supervised by their assigned mentee (one of the assistants of P. Gonçalves, N. Krouglova and G. Moss). The direct involvement in a time-limited project and the immediate access to guidance and expertise yielded a more long-lasting result than a passive lecture series. Although each student group concentrated on a specific problem, they all benefited from exposure to the other projects during the final presentation of the results; in this manner, the students were not only proficient in their own project but knowledgeable in related topics.

Thus, with this format in mind, the goals of our workshop were as follows: *i)* expose students to the ever-growing field of SBI, *ii)* foster networking between the students by arranging them in groups and giving them state-of-the-art research projects to complete and *iii)* present the participants an opportunity to meet and discuss with experts in the field.

After receiving feedback from both iterations of this workshop, we are confident we have achieved all these goals. In future iterations, this workshop can establish itself as a significant opportunity for young scientists to study problems at the intersection of neuroscience and machine learning. By providing specialized training opportunities and by encouraging the formation of collegial networks and connections, we hope to encourage, educate and foster the next generation of scientists in these interesting and important fields.

2 Topic of the workshop: Simulation-based inference

Deep learning (DL) and artificial neural networks (ANNs) dominate many facets of human society. From image recognition (Pak and Kim, 2017), natural language processing (Otter et al., 2020; Brown et al., 2020) to competitive games (Silver et al., 2017) and protein folding (Ruff and Pappu, 2021), DL has become an increasingly important feature of the algorithmic toolbox.

All these breakthroughs are based on the inherent ability of ANNs to effectively learn functional mappings between elements, be it images, time series or biological data. SBI (Cranmer et al., 2020) leverages this capability to derive statistical inferences from complex models that are difficult or impossible to perform using traditional methods. Its efficacy means that it has found use in fields such as physics (Vallisneri et al., 2024), biology (Zaballa and Hui, 2023), and economics (Dyer et al., 2022), where models can be too complex for analytical solutions or traditional parameter fitting approaches. Instead of relying on explicit likelihood functions, SBI employs simulations of pre-determined models to understand the behavior of the system. In this context, ANNs, particularly deep learning techniques, are often used to approximate the posterior distributions of model parameters given observed data.

The process of SBI typically involves three main steps:

1. Simulation: Generate synthetic data by running simulations of the model with different parameter values. This helps create a comprehensive dataset that captures the range of possible behaviors of the model.

2. Learning: Use ANNs to learn the relationship between the simulated data and the parameters. Techniques like Approximate Bayesian Computation (ABC) or Neural Posterior Estimation (NPE) are employed to approximate the posterior distribution. ANNs excel here due to their capacity to handle high-dimensional data and complex, non-linear relationships.

3. Inference: Apply the trained model to real observed data to infer the underlying parameters or make predictions. The learned neural network can provide posterior samples or estimates of the parameters, giving insights into the system's behavior and uncertainty quantification.

SBI has several advantages. It is flexible and can be applied to models for which the likelihood is intractable or expensive to compute. It also accommodates high-dimensional data and complex dependencies between variables. By using ANNs, SBI can effectively manage large-scale simulations and provide accurate and interpretable inferences. Overall, SBI represents a robust and versatile framework that harnesses the power of simulations and neural networks to tackle complex inference problems across various scientific disciplines.

Despite its potential, SBI often requires significant computational resources, particularly in the number of (potentially computationally expensive) model simulations that may be necessary to infer the likelihood or the posterior distribution. This issue is particularly acute when the number of model parameters to infer or the complexity of the underlying model increases.

As a leading expert in SBI, Prof. Gonçalves was the natural choice to lead this year's iteration of the "Bio-inspired deep learning" workshop (Boelts et al., 2019; Gonçalves et al., 2020; Ramesh et al., 2022; Deistler et al., 2022). Taking advantage of today's ever-increasing capacity to generate data at unprecedented scales and resolutions and using modern machine learning methods, Prof. Gonçalves has pushed the development of SBI algorithms (Tejero-Cantero et al., 2020) (including novel sequential neural posterior estimators, e.g., SNPE-B, Lueckmann et al., 2017) that have successfully been used to extract accurate parameter estimates of interpretable theoretical models in a scalable fashion. He is currently applying these methodologies with his doctoral student, N. Krouglova, to extract mechanistic insights from neural circuit data. Additionally, the teaching assistant G. Moss (a member of the lab where Prof. Gonçalves undertook a Postdoctoral stay) applies SBI to study evolution of ice shelves (Moss et al., 2023).

Given these expertise, four projects were presented to the student groups:

- **Mining Silver**: In this project, inspired by the work of (Brehmer et al., 2020), rather than treating the simulator as a blackbox (i.e., without any knowledge of gradients or intermediate

latent variables), the group took advantage of the knowledge of intermediate variables of the simulator and tested whether this information improved the efficiency of SBI.

- **Benchmarking SBI**: (Lueckmann et al., 2021) is a benchmarking study comparing a range of SBI algorithms, simulators and metrics. The goal of this project was (a) to extend the benchmarking study with a simulator with a high-dimensional output – a regime that has not been extensively tested within the SBI toolbox –, (b) to expose the students to the advantages and pitfalls of different SBI methods and (c) to provide inspiration for future developments in this challenging simulator regime.

- **Amortised Regression**: The goal of this project was to explore the possibility of using SBI to run a Machine Learning (ML) algorithm in an amortised fashion, i.e., to have a trained neural density estimator to instantaneously run the algorithm on arbitrary new data. The goal was then to compare the performance of this method with standard off-the-shelf Bayesian methods. A very related – and perhaps more sophisticated version – of this idea has been developed by (Hollmann et al., 2023).

- **Summary statistics**: Summary statistics are statistics that aim to characterize a given dataset. However, it is currently unclear how to extract maximally informative summary statistics about the model parameters, while ensuring efficient and accurate inference. The goal of this project was to compare the SBI performance given different strategies to compute summary statistics (expert-designed, variance-preserving, embedding-nets).

While all the projects were ambitious in scope and topic, the results achieved by the students are nonetheless impressive and point to interesting future endeavours. We are grateful for all their hard work and openness to work on novel topics.

References

Jan Boelts, Jan-Matthis Lueckmann, Pedro J Goncalves, Henning Sprekeler, and Jakob H Macke. Comparing neural simulations by neural density estimation. In *2019 Conference on Cognitive Computational Neuroscience. Berlin, Germany: Cognitive Computational Neuroscience*, pages 578–81, 2019.

Johann Brehmer, Gilles Louppe, Juan Pavez, and Kyle Cranmer. Mining gold from implicit models to improve likelihood-free inference. *Proceedings of the National Academy of Sciences*, 117(10): 5242–5249, February 2020. ISSN 1091-6490. doi: 10.1073/pnas.1915980117. URL http://dx.doi.org/10.1073/pnas.1915980117.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

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

Michael Deistler, Pedro J Goncalves, and Jakob H Macke. Truncated proposals for scalable and hassle-free simulation-based inference. *Advances in Neural Information Processing Systems*, 35: 23135–23149, 2022.

Joel Dyer, Patrick Cannon, J Doyne Farmer, and Sebastian Schmon. Black-box bayesian inference for economic agent-based models. *arXiv preprint arXiv:2202.00625*, 2022.

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

Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. Tabpfn: A transformer that solves small tabular classification problems in a second, 2023.

Jan-Matthis Lueckmann, Pedro J Goncalves, Giacomo Bassetto, Kaan Öcal, Marcel Nonnenmacher, and Jakob H Macke. Flexible statistical inference for mechanistic models of neural dynamics. *Advances in neural information processing systems*, 30, 2017.

Jan-Matthis Lueckmann, Jan Boelts, David S. Greenberg, Pedro J. Gonçalves, and Jakob H. Macke. Benchmarking simulation-based inference, 2021.

Guy Moss, Vjeran Višnjević, Olaf Eisen, Falk M Oraschewski, Cornelius Schröder, Jakob H Macke, and Reinhard Drews. Simulation-based inference of surface accumulation and basal melt rates of an antarctic ice shelf from isochronal layers. *arXiv preprint arXiv:2312.02997*, 2023.

Daniel W Otter, Julian R Medina, and Jugal K Kalita. A survey of the usages of deep learning for natural language processing. *IEEE transactions on neural networks and learning systems*, 32(2): 604–624, 2020.

Myeongsuk Pak and Sanghoon Kim. A review of deep learning in image recognition. In *2017 4th international conference on computer applications and information processing technology (CAIPT)*, pages 1–3. IEEE, 2017.

Poornima Ramesh, Jan-Matthis Lueckmann, Jan Boelts, Álvaro Tejero-Cantero, David S Greenberg, Pedro J Gonçalves, and Jakob H Macke. Gatsbi: Generative adversarial training for simulation-based inference. *arXiv preprint arXiv:2203.06481*, 2022.

Kiersten M Ruff and Rohit V Pappu. Alphafold and implications for intrinsically disordered proteins. *Journal of Molecular Biology*, 433(20):167208, 2021.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017.

Alvaro Tejero-Cantero, Jan Boelts, Michael Deistler, Jan-Matthis Lueckmann, Conor Durkan, Pedro J Gonçalves, David S Greenberg, and Jakob H Macke. Sbi–a toolkit for simulation-based inference. *arXiv preprint arXiv:2007.09114*, 2020.

Michele Vallisneri, Marco Crisostomi, Aaron D Johnson, and Patrick M Meyers. Rapid parameter estimation for pulsar-timing-array datasets with variational inference and normalizing flows. *arXiv preprint arXiv:2405.08857*, 2024.

Vincent Zaballa and Elliot Hui. Approximation of intractable likelihood functions in systems biology via normalizing flows. In *NeurIPS 2023 Generative AI and Biology (GenBio) Workshop*, 2023.

Mining Silver

Grégoire Degobert-Yasmine[1], Javier Ferri[2], Pablo Rojas[3], Su Saka[4]

Abstract

Simulation-based inference methods (SBI) are Bayesian inference methods used to infer the parameters of a simulator from observed data. The simulator is typically assumed to be blackbox in which latent variables are not easily accessible. Here, we challenged this idea by performing Sequential Neural Posterior Estimation using the latent variables of a modified "Two Moons" task and an advection-diffusion model. We show that faster convergence of the posterior estimate to a ground truth was achieved when latent variables were used. This was particularly clear for the "Two Moons" task and less clear for the advection-diffusion model. Collectively, our results show that accessing latent variables from the simulator may result in an improved performance of SBI.

1 Introduction

To understand complex phenomena of nature, scientists use computer simulations that generate synthetic data from models with predefined parameter values. However, sometimes the inverse problem –given some observations find the combination of parameters of the model that explains the observations– is more interesting since it can yield mechanistic insights about the system under study. This is quantified through the posterior probability, which generally is not computable and requires alternative parameter inference methods collectively known as simulation-based inference methods. Simulation-based inference (SBI) has been used in different fields of natural sciences such as physics (1), epidemiology (2), molecular biology (3), and neuroscience (4), and has grown rapidly in popularity with the development of new deep-learning algorithms, allowing training neural networks with data from the simulator to infer parameters from high-dimensional data.

In most applications, the simulator used in SBI is taken as a blackbox. It is possible to sample observations from a posterior distribution, but intermediate latent variables are not accessible. This assumption was challenged in Brehmer et al. (5), where authors successfully used latent gradients of the simulator to improve the efficiency of a neural network that estimates the likelihood. In this workshop, we were given a project to push further the efficiency of SBI by using latent variables of the simulator instead of latent gradients. Latent variables are supposedly simpler distributions than the likelihood and should be easier to learn by a network. To explore this idea, we started by accessing the latent variables of the "Two Moons" task defined in Greenberg et al. (6), and estimated the likelihood using sequential neural posterior estimation (SNPE).

[1]Ecole Normale Supérieure, Paris, France
[2]Max Planck Institute for Brain Research, Frankfurt am Main, Germany
[3]University of Kassel, Kassel, Germany
[4]Center for Neurogenomics and Cognitive Research, Vrije Universiteit Amsterdam, Amsterdam, Netherlands

2 Methods

2.1 "Two Moons" model

The "Two Moons" task, introduced in Greenberg et al. (6), consists in a toy synthetic model in which the inference of parameters is considerably simplified when considering the latent variables. In the Two Moons, a bidimensional observation x is sampled from a likelihood with a bidimensional parameter θ such as:

$$x|\theta = \begin{bmatrix} r\cos\alpha + 0.25 \\ r\sin\alpha \end{bmatrix} + \begin{bmatrix} |-\theta_1+\theta_2|/\sqrt{2}+\epsilon \\ (-\theta_1+\theta_2)/\sqrt{2}+\epsilon \end{bmatrix} \tag{1}$$

where $r \sim \mathcal{N}(0.1, 0.01^2), \alpha \sim \mathcal{U}(-\pi/2, \pi/2)$ and $\epsilon \sim \mathcal{N}(0, 0.1^2)$ and the latent variable is:

$$z = \begin{bmatrix} |-\theta_1+\theta_2|/\sqrt{2}+\epsilon \\ (-\theta_1+\theta_2)/\sqrt{2}+\epsilon \end{bmatrix} \tag{2}$$

2.2 Advection-diffusion model

In order to test that accessing latent variables improves likelihood estimation in a physically more realistic problem, we designed a synthetic generic example which mimics the transport of substances over continuum media (for instance, a liquid) in a fictitious experiment. We used the advection-diffusion equations in which the a certain substance whose concentration $C(x,t)$ diffuses and is advected in a medium, and the measurement is taken as the total amount of the substance measured in the readout zone at a certain distance d_{wall} from the starting point (See Figure 1). The partial differential equation (PDE) for the model describe the change in concentration $C(x,t)$ depending on space and time as:

$$\begin{cases} \dfrac{\partial C}{\partial t} = D\dfrac{\partial^2 C}{\partial x^2} - v\dfrac{\partial C}{\partial x} & \forall x \text{ in } (-\infty,\infty) \qquad \forall t \text{ in } [0,\infty) \\ C(x,0) = \delta(0) \end{cases} \tag{3}$$

where D is the *diffusion coefficient* and v is the *advection speed* or *drift velocity*. Equation 3 can be solved analytically:

$$C(x,t) = \frac{1}{\sqrt{4\pi D t}}\exp(-\frac{x^2}{4Dt}) + vt \tag{4}$$

The solution can be interpreted as a Gaussian whose variance σ^2 increases linearly in time as $\sigma^2 = 2Dt$, and the mean value is transported at constant speed as $\mu = vt$. Measurements P depend on both the parameters of the model D and v, as can be seen in Figure 2.

For the purposes of the simulator, we can dispense with solving the PDE numerically, as we can represent the state of the system at time $t = t_{measurement}$ with σ and μ. The measurements can also be computed analytically, as the integral of $C(x, t_{measurement})$ from the distance d_{wall} up to ∞:

$$P = \int_{d_{wall}}^{\infty} C(x, t_{measurement})dx = \frac{1}{2}(1 - \texttt{erf}\left(\frac{d_{wall}-\mu}{\sigma}\right)) \tag{5}$$

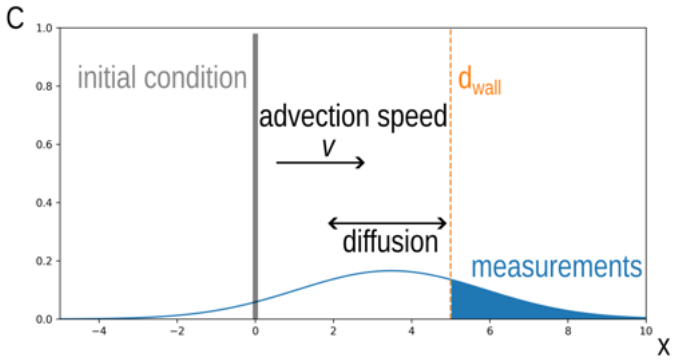


Figure 1: Scheme of the diffusion model. A substance described by its concentration $C(x,t)$ diffusion over a medium, which in turn is subjected to advected transport with speed v (Equation 3). In the fictitious experiment represented by the model, measurements are taken as the amount of substance that is adsorbed by an instrument located at a distance d_{wall}. The task is to infer possible values of the parameters defining the physical problem D (diffusion coefficient) and v (advection speed or drift velocity) from the measurements.

where `erf()` is the *error function*. In our pipeline, the simulator takes parameters $D \in [0,5]$ and $v \in [0,6]$ to generate measurements P through the computation of latent variables μ and σ, considering the fixed hyperparameters $d_{wall} = 5$ and $t_{measurement} = 1$.

We also considered a variant of the diffusion model that is not purely deterministic, but is subjected to stochastic noise in both the computation of the latent variables and the output, following:

$$\begin{aligned} \mu &= vt + \mathcal{N}(0, 0.1^2) \\ \sigma &= \sqrt{2Dt} + \mathcal{N}(0, 0.1^2) \\ P &= \frac{1}{2}(1 - \texttt{erf}\left(\frac{d_{wall} - \mu}{\sigma}\right)) + \mathcal{N}(0, 0.01^2) \end{aligned} \tag{6}$$

2.3 Sequential neural posterior estimation

We employed sequential neural posterior estimation (SNPE) to infer the posterior distribution of our models' parameters. This is a Bayesian inference technique that takes advantage of neural networks, particularly suited for scenarios where the likelihood function is computationally expensive or unknown.

First, we constructed a simulator for either the "Two-Moons" task or the advection-diffusion model to simulate a large amount of data (x) from parameters (θ) sampled from the prior. We then used both x and θ to train the density estimator with the SNPE algorithm. For this, we used the sbi package from Tejero-Cantero et al. (7), using Neural Spline Flow (8) as the density estimator. This algorithm iteratively improves the density estimator by sequentially adding simulated data and its corresponding parameters, learning the mapping from parameters to

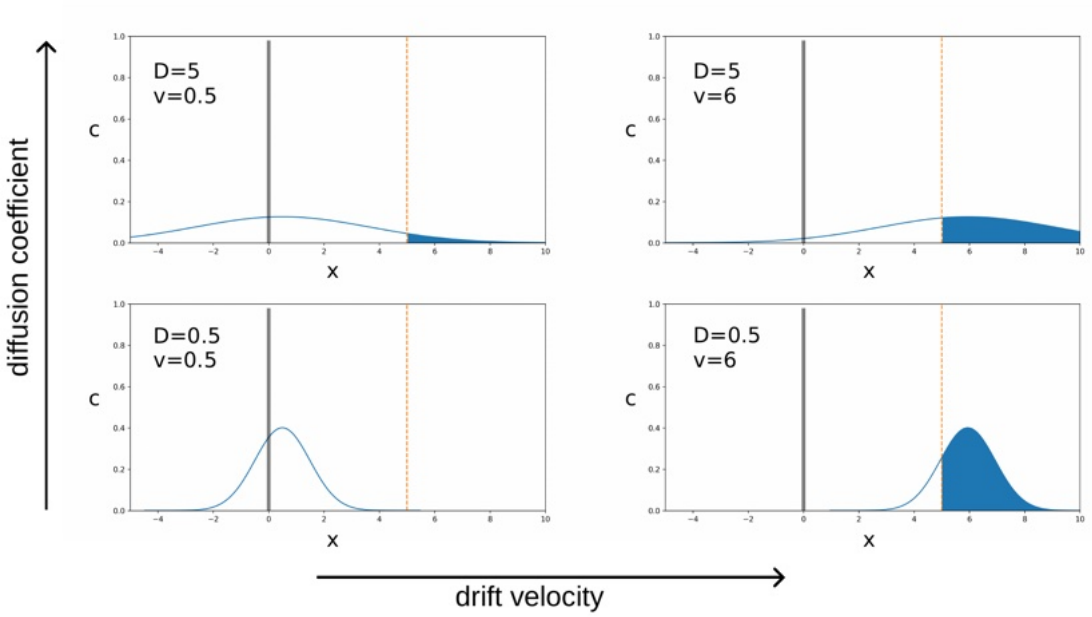


Figure 2: Examples of observation P in the fictitious experiment of the advection-diffusion model, as a result of the interplay between the diffusion coefficient D and drift velocity v. Larger v generates larger P, but the influence of D depends on the value of v. This interplay generates a redundancy in the output with respect to the parameters, where multiple combinations of D and v generate the same observation P.

data. We then obtained the posterior distribution ($p(\theta|x)$) from the trained density estimator (Fig 3b).

We conducted Simulation-Based Calibration (SBC) to validate the inferred posterior distribution. With the prior distribution, we generated ground truth parameter samples and simulated observations to infer a posterior from each of them. Then, a group of samples from each posterior were generated and ranked according to how many were below or above the ground truth parameter.

To calculate $p(\theta|x)$ through a latent variable, we estimated $p(z|x)$ and $p(\theta|z,x)$ separately as described above, and then we use them to sample from $p(\theta|x)$. Note that the posterior distribution can be accessed as follows:

$$p(\theta|x) = \int p(\theta, z|x)dz = \int p(\theta|x,z)p(z|x)dz \tag{7}$$

2.4 Comparison of "classical" SNPE and two-step SNPE

In order to compare the performances of the two SNPE method, we defined empirical distances between posterior density estimators. As the neural density estimators are amortized, i.e., they provide an estimation of the posterior distribution for any observation x_o (assuming there exists parameters θ_o in the prior that allow to generate x_o), we measured distances between estimated posterior distributions for different values of observations, thus giving an average

distance between neural network estimated distributions. We measured the distances between distributions by using two empirical metrics on distribution samples. The first one is sliced Wasserstein distance (9), which computes the empirical mean of Wasserstein distances between one-dimensional linear projections of the distributions. We thus define the first metric as follows:

$$\frac{1}{m}\sum_{i=1}^{m} SW(\hat{P}^n_{x_i}, \hat{Q}^n_{x_i}) \tag{8}$$

where $(x_i)_{1\leq i\leq m}$ are the different observations, $\hat{P}^n_{x_i}$ and $\hat{Q}^n_{x_i}$ are sets of n samples from the two posterior estimators evaluated in x_i, and SW is the p-order sliced-wasserstein distance:

$$SW(\hat{P}, \hat{Q}) = \left(\frac{1}{l}\sum_{k=1}^{l}\left(W_p\left(\pi^k_{\hat{P}}, \pi^k_{\hat{Q}}\right)\right)^p\right)^{\frac{1}{p}}$$

where $\pi^k_{\hat{P}}$ and $\pi^k_{\hat{Q}}$ are projections of the samples on a k-th random axis of the d-dimensional unit sphere (with d is the dimension of the distributions) and W_p is the p-order, one dimensional Wasserstein distance. In practice we used $p = 2$ and $l = 50$.

The second metric is obtained from an empirical method called classifier two-sample test (C2ST) (10), which relies on training a classifier to distinguish samples drawn from supposedly two distributions P and Q. If the N-fold cross-validation accuracy of the classifier is close to 0.5, the two sets of samples can be considered as drawn from the same distributions while an accuracy close to 1 suggests that the two distributions have significant differences that are perceived by the classifier. This allows us to introduce our second metric as the mean C2ST accuracy of the classifier for distributions of two neural posteriors estimated for different observations.

$$C2ST(\hat{P}, \hat{Q}) = \frac{1}{m}\sum_{i=1}^{m}\mathcal{A}_N\left(\hat{P}^n_{x_i}, \hat{Q}^n_{x_i}\right)$$

where $\mathcal{A}(P, Q)$ is the N-fold cross validation accuracy of a random-forest classifier at discriminating samples drawn from two sets P and Q. In practice we used $N = 5$.

For both metrics we used ten observations values obtained by simulation from $m = 10$ random parameters samples from the prior parameter distribution, and drawn $n = 2000$ samples from each evaluated posterior distribution. However, we believe that increasing the number of observations and samples, while being computationally costly, would increase the consistency of our results.

3 Results

3.1 SPNE on "Two Moons"

We first performed SNPE in the "Two Moons" task from Greenberg et al. (6), both directly and through accessing a latent variable (Fig 3). We could successfully estimate the posterior with no distinction in the SBC validation for both cases (Fig 3c). Thus, this demonstrated that by using a latent variable, the same results can be obtained as by estimating directly $p(\theta|x)$.

Visually, we see that the inferer trained with the two-step neural posterior estimation achieves a high quality inference of the posterior distribution for a lower simulation budget (1000 simulations) than the inferer trained with the standard method. However, we see that with the same number of samples, a much more homogeneous distribution can be plotted with the standard method. This is easily explained because we sample much less values from each intermediary distributions than we sample in total, resulting in an irregularly shaped distribution 4.

In order to better quantify the performances of the two posterior estimation methods, we computed the distances between the distributions using the empirical sliced-Wasserstein distance and C2ST metrics. We computed the average distances between estimators depending on the number of simulations (figure 5, bottom), showing that both methods converge towards the same distribution.

We then evaluated the distances of the neural estimators to a "ground truth" estimator (figure 5, top/middle). Such an estimator has been obtained by training a Neural Posterior Estimator with a very high number of simulations (50000) with the two methods, that we considered being an almost perfect posterior estimator for any observation. Given that both methods converge toward similar results, any of the two estimators could be used as a ground truth.

The average sliced wasserstein distance between the two ground-truth distributions gives us a significance limit. Unfortunately, the differences between distances to ground truth for both methods is never higher than the differences between ground-truth. The c2st metric suggests that in average, the twostep NPE method converges faster toward a high accuracy estimation of the posterior, outperforming the standard method in the 200-1000 simulations range. However, due to the sampling process, the two-step NPE requires a larger number of samples to correctly approximate the distribution, resulting in larger distance estimations for large numbers of simulations for two-step NPE than for the classical method. However we believe that such weakness could be attenuated with larger number of latent variable samples and lower number of θ samples per latent variable sample.

3.2 SNPE on advection-diffusion model

Analogue to the "Two Moons", we performed SNPE in the advection-diffusion task described in subsection 2.2 in both scenarios: direct inference and with access to the latent variables.

We first performed direct inference to verify that the inferred parameters capture the physical intuition we described in subsection 2.2, i.e. the redundancy in the parameters D and v with respect to the observations P. For instance, in Figure 2 we observed for small values of D and v the transport is slow and the observed $P \approx 0$ at $t_{measurement}$. This is captured by the direct inference with SNPE (Figure 6), as an entire region of the plane $v - D$ close to the axis is inferred as solutions of the inverse problem, even in the case of purely deterministic simulations.

The case of finite observations P also report redundancy in the parameters. In Figure 7 we observe the posterior probability in the plane $v - D$ that is inferred to explain the observations generated with two sets of parameters, both yielding the same observations. As the regions of higher probability are reproduced in both cases, we are able to recapture the redundancy in parameters. The effect of stochastic noise in the direct inference is to "widen" the posterior

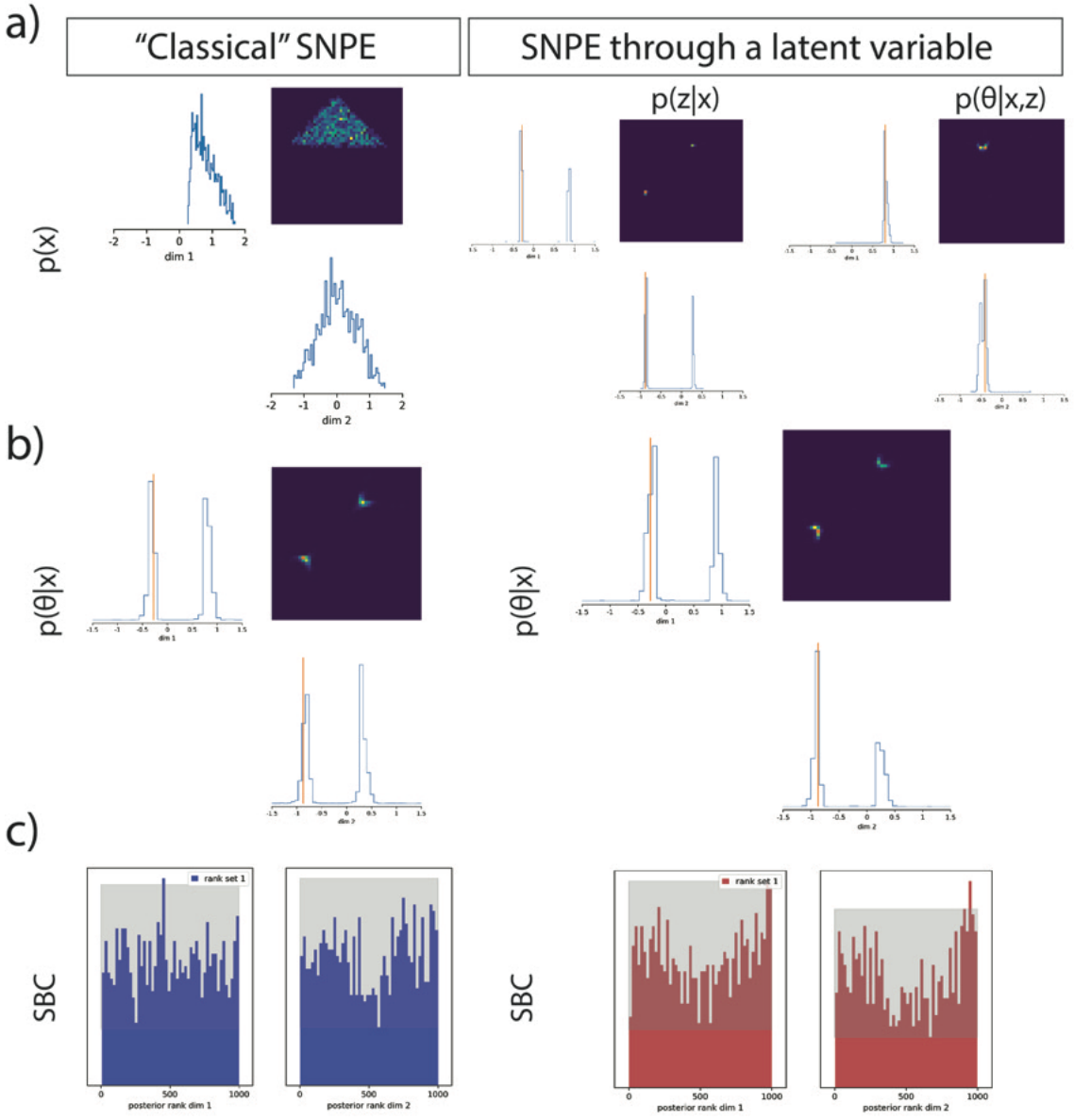


Figure 3: Posterior estimation through SNPE for the "Two Moons" task. a) Left: Classical SNPE ($\mathrm{p}(x)$). Right: SNPE through a simulator latent variable ($\mathrm{p}(z|x)$ and $\mathrm{p}(\theta|x,z)$). b) Estimation of $\mathrm{p}(\theta|x)$. c) SBC validation. Both methods show a uniform distribution of ranks, meaning the posterior was successfully estimated.

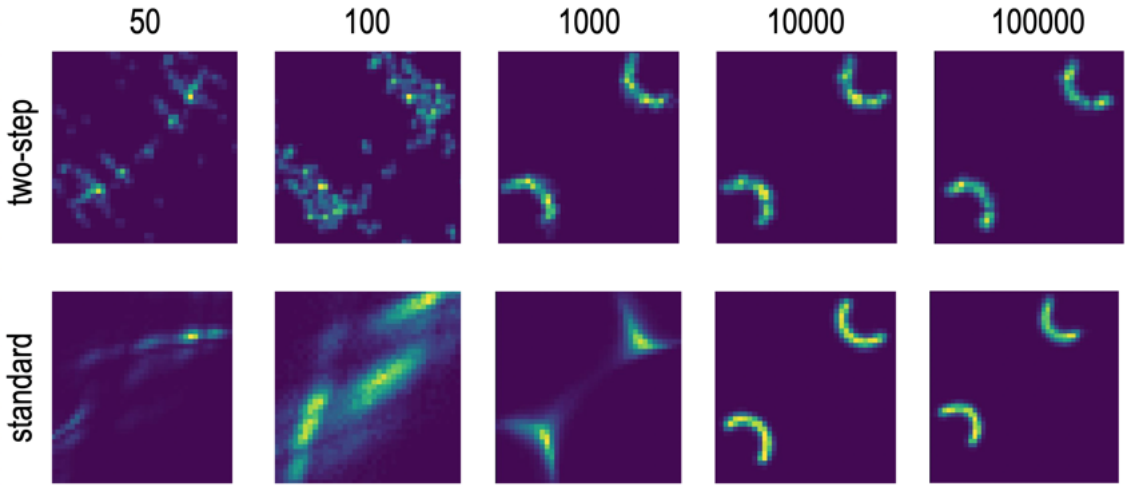


Figure 4: 2D distributions of the estimated posterior distribution of θ for an arbitrary value of x_o by two-step (upper) and standard (lower) neural posterior inferers for various simulations budgets (indicated on top of the square plots). 2D histograms were plotted with 100000 samples for the standard method, and 1000 samples of θ for each z samples, with 1000 z samples, for the two-step method, thus resulting in the same total number of samples for the two methods

.

distribution, as can be seen in Figure 8, which shows a comparison for the inferred posterior distribution for the same set of parameters in two cases: purely deterministic and with added stochastic noise.

As outlined previously, in order to compare the performances of the two SNPE methods, we used various number of simulations for inference using both methods. We then evaluated the difference between distribution samples for various numbers of simulations to a ground truth distribution. The ground truth distribution was created by running a relatively high number of simulations for the inference. For the diffusion model , this was created by performing SNPE on 100000 simulations. For the comparisons we used ground truths created via both methods.

Surprisingly, the comparison of the sliced Wasserstein distance and C2ST accuracy between distributions obtained via simulations of various numbers and the ground truth did not reveal a clear pattern of earlier convergence of the likelihood to a ground truth (Figure 9).

4 Conclusions

Here, we explored if utilizing the latent variables of the simulator for inference would improve the efficiency of SNPE. Focusing on a modified version of the "Two Moons" task defined in Greenberg et al. ([6]), we achieved convergence of the likelihood to a ground truth with fewer simulations when utilizing latent variables in a two-step method compared to standard SNPE. Qualitatively, the number of necessary simulations to obstain a distribution visually similar to ground truth was decreased almost 10-fold for the two-step method, showing increased efficiency of inference. However, we were not able to illustrate this significant increase in performance with a quantitative metric. We believe that such difficulty to quantify performance of estimators

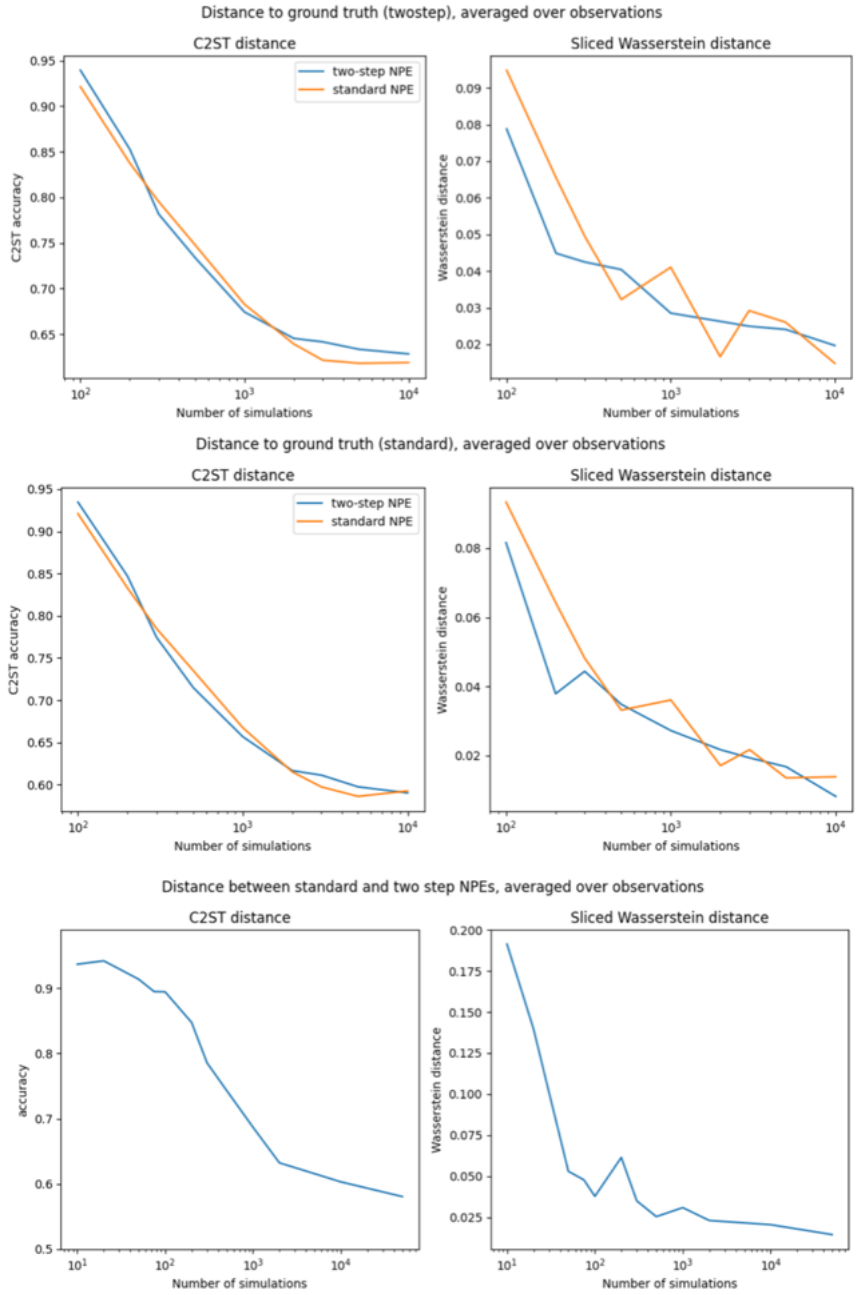


Figure 5: Distances of the two posterior 50000-simulation-ground-truth estimators obtained with two-step simulations (top), standard (middle) and distances between estimators trained with the two-step and standard neural posterior estimation methods.

.

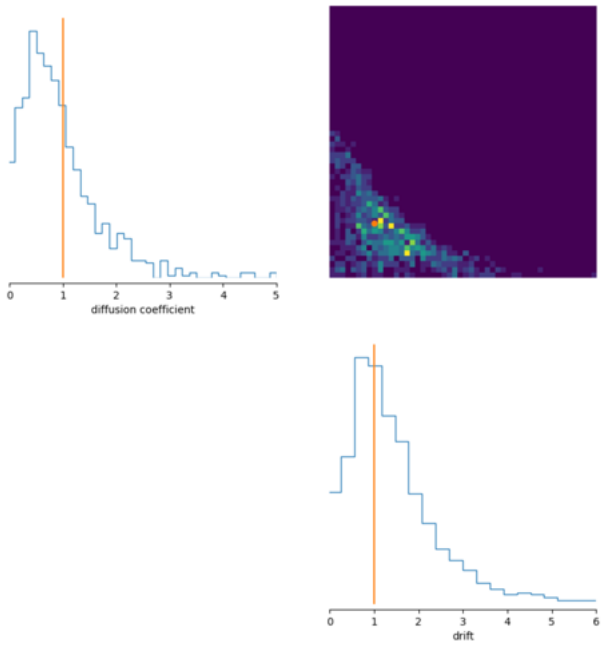


Figure 6: Small values of D and v (in this case $D = 1$ and $v = 1$) generate observations numerically close to $P \approx 0$ (see Figure 2), and therefore are inferred as equivalent in the deterministic simulations.

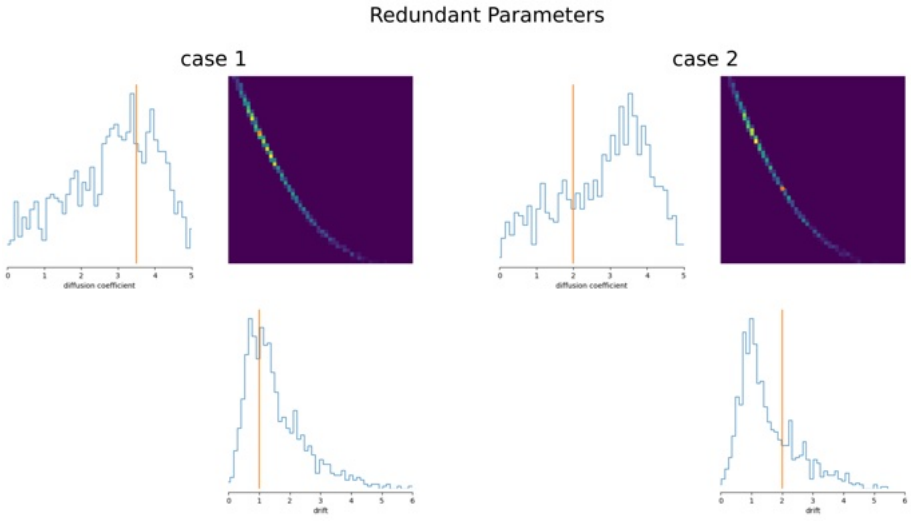


Figure 7: The inference with SNPE reports regions with similar probability in the advection-diffusion model, and two set of parameters corresponding to the same region produce same observations and yield similar regions. This is a confirmation of the redundancy intuited from physical reasoning. The simulations for this example are performed with the deterministic version.

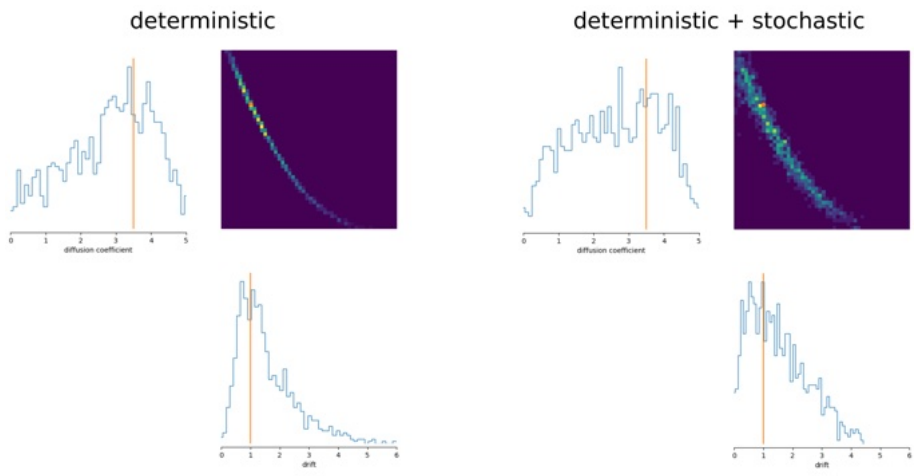


Figure 8: Comparison of the posterior distribution inferred with the purely deterministic model (left panel) and deterministic model with stochastic noise (right panel) for the same parameters.

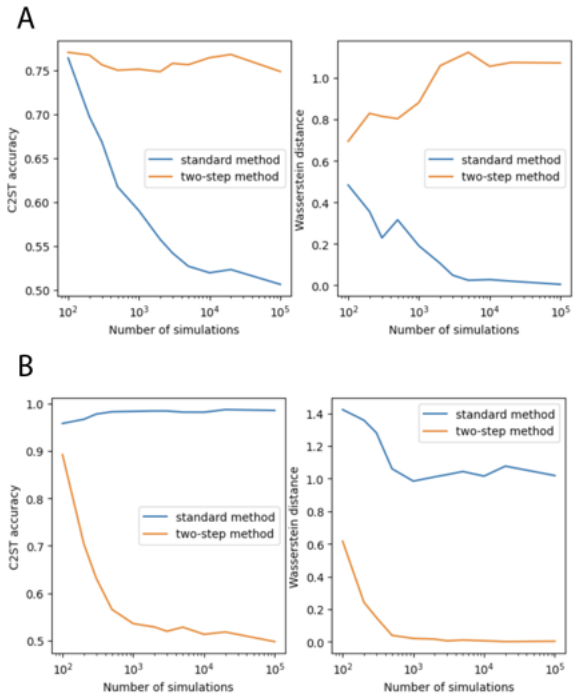


Figure 9: The difference between posterior distribution inferred via either direct inference (standard method) or inference with access to latent variable (two-step method) to a ground truth distribution obtained via direct inference (A) or via inference with access to latent variable (B).

is due to the sampling bias of the two-step inference method and to the cost of the empirical distance in terms of samples and thus computation. It might be possible that in models with a larger number of sequentially generated latent variables, the two-step estimation method could reveal its full potential by learning several simple distributions instead of a very complex one.

To further confirm our initial results, we applied the same idea to an advection-diffusion model with two parameters (diffusion coefficient and drift velocity) and concentration measurements depending on space and time, measured at a wall at an arbitrarily defined distance from the starting point. However, here we did not observe a increase in efficiency. It is likely that the structure of the advection-diffusion equations is not the ideal testbed for the method. Intuitively, a transport problem subjected to a highly nonlinear measurement would appear as a problem whose decomposition into smaller problem would improve its inference. However, the comparison with the case of the "Two Moons" problem lead us to the speculation that the multimodal posterior distribution, along with the how relatively complex or simple are the maps from the parameter space to the latent variable space and from the latent variable space to the observations, may play a role in how much the inference efficiency is improved.

Our results show that using latent variables of a simulator to estimate the likelihood could potentially lead to an improvement of the efficiency of SBI. However, further analyses are needed to clarify exactly to which point and how.

5 Acknowledgements

We are thankful to the organizers of the Workshop fur such as smooth planification, to the faculty for their insightful lectures and explanations, and to all the participants for the magnificent experience during the Workshop.

References

[1] Georges Aad, Tatevik Abajyan, Brad Abbott, Jalal Abdallah, S Abdel Khalek, Ahmed Ali Abdelalim, R Aben, B Abi, M Abolins, OS AbouZeid, et al. Observation of a new particle in the search for the standard model higgs boson with the atlas detector at the lhc. *Physics Letters B*, 716(1):1–29, 2012.

[2] Trevelyan J McKinley, Joshua V Ross, Rob Deardon, and Alex R Cook. Simulation-based bayesian inference for epidemic models. *Computational Statistics & Data Analysis*, 71: 434–447, 2014.

[3] Hippolyte Verdier, François Laurent, Alhassan Cassé, Christian L Vestergaard, Christian G Specht, and Jean-Baptiste Masson. Simulation-based inference for non-parametric statistical comparison of biomolecule dynamics. *PLoS Computational Biology*, 19(2):e1010088, 2023.

[4] Jan Boelts, Philipp Harth, Richard Gao, Daniel Udvary, Felipe Yáñez, Daniel Baum, Hans-Christian Hege, Marcel Oberlaender, and Jakob H Macke. Simulation-based inference

for efficient identification of generative models in computational connectomics. *PLOS Computational Biology*, 19(9):e1011406, 2023.

[5] Johann Brehmer, Gilles Louppe, Juan Pavez, and Kyle Cranmer. Mining gold from implicit models to improve likelihood-free inference. *Proceedings of the National Academy of Sciences*, 117(10):5242–5249, 2020.

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

[7] Alvaro Tejero-Cantero, Jan Boelts, Michael Deistler, Jan-Matthis Lueckmann, Conor Durkan, Pedro J Gonçalves, David S Greenberg, and Jakob H Macke. Sbi–a toolkit for simulation-based inference. *arXiv preprint arXiv:2007.09114*, 2020.

[8] Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural spline flows. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[9] Soheil Kolouri, Yang Zou, and Gustavo K Rohde. Sliced-wasserstein kernels for probability distributions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[10] David Lopez-Paz and Maxime Oquab. Revisiting classifier two-sample tests, 2018.

Benchmarking Neural estimation methods

Lordstrong Akano[1], Nicoleta Contruz[2], Bharat Joshi[3], Janko Petkovic[4],

Abstract

With the advancement of both technical and formal instruments, richer datasets and more refined theories are being introduced to the scientific community in almost every field. This increased model complexity and data availability, however, has made parameter inference progressively challenging, rendering it by all rights one of the "hard problems" of applied research. The evaluation of the parameter posterior, in particular, has become a particularly barren endeavor, since what models have gained refinement had to be paid, sadly, in mathematical accessibility. To tackle this issue, as set of new techniques pertaining to the field of *simulation based inference*, has recently gained attention. These techniques, able to leverage the increasing power of state of the art computational systems, are able to bypass mathematical intractability via neural network driven approximation and Monte Carlo sampling. In this work, we focus on studying a set of these techniques, evaluating their inference performance on a well known ecological problem. In particular, we explore the dependence of the inference accuracy on the training set dimensions, on possible summary statistics and formulate some hypothesis about the observed advantages and pitfalls of SBI methods.

1 Introduction

As advancements in Neurotechnology continue to improve neural recording techniques, the size of neural data available for analyses and inference continue to exponentially increase. This increase necessitates the development and validation of models. As in other scientific fields, models in Neuroscience have parameters which are often tuned to fit experimental observations.

Parameter inference is a scientific problem which gets increasingly difficult with increasingly complex, stochastic models. Many times being mathematically intractable. Simulation-based inference (SBI) employs a Bayesian framework to estimate parameters of a model. It is still, however, limited by mathematical intractability in many cases.

Recent methods get around this intractability by utilizing neural networks (exploiting their utility as universal function approximators) to learn mappings between the output of models (synthetic data) and its parameters, minimizing the difference between the generated data and the experimental/observed data (ground truth). Therefore giving access to the posterior. These class of methods in simulation-based inference are referred to as "Neural estimation" methods.

In this workshop project, we benchmarked two of such methods; Neural posterior estimation and Neural likelihood estimation, using a toy diffusion model of 3 populations. Attempting to infer the known parameters of the model, given images of the steady-state intermixed populations.

[1]Institute of Science and Technology, Austria
[2]Institute of Science and Technology, Austria
[3]Institute of Medical Microbiology and Hygiene, University Medical Center, Mainz,
[4]Institute of Experimental Epileptology and Cognition Research, University of Bonn

2 Methods

2.1 Benchmark problem

The performance of the simulation-based inference methods is validated on the "biological Rock-Scissors-Paper" (RPS) toy model

$$\begin{cases} \partial_t a = D\,\nabla^2 a + \mu\, a\,(1-\rho) - \sigma\, a\, c \\ \partial_t b = D\,\nabla^2 b + \mu\, b\,(1-\rho) - \sigma\, b\, a \\ \partial_t c = D\,\nabla^2 c + \mu\, c\,(1-\rho) - \sigma\, c\, b \end{cases} \tag{1}$$

with $a, b, c : \mathbb{R}^2 \times \mathbb{R} \to \mathbb{R}$ and $\rho = \min\{1, a+b+c\}$. This model has been widely studied in ecology to describe the interaction of three different mobile species in cyclic-dominance with each other. Given a population vector $x = (a(t_0), b(t_0), c(t_0))$, this work aims at exploring the ability of the different SBI methods to infer the posterior distribution $f(\theta|X)$ of the parameter vector $\theta = (\mu, \sigma, D)$ used to generate a specific instance $x = x(\theta)$ of the system (1).

2.2 System simulation

The three populations are initialized at an equal, constant value across the whole spatial domain corresponding to one of the system's unstable fixed points. Symmetry break is then introduced adding normally distributed noise (0 mean and 0.01 standard deviation), and finally the system is evolved for 100 time steps using a FTCS numerical scheme with a first order Euler integration method with a dt of 1.0. The result is then saved together with the parameter vector used to generate it. This procedure is carried out 10^5 times, effectively generating the ground truth joint distribution $f(x, \theta)$.

2.3 Inference model training

A set of subsamplings with numerosities $n_{sub} = (500, 1000, 5000, 10000, 50000)$ is carried out on the ground truth joint, and the obtained subsamples are used to train the neural inference methods. In this fashion, it is possible to study the inference methods' power in relation to the training set it is provided with. For all the neural models, the training minibatch size was taken to be 256, and the number of training epochs was automatically chosen by the SBI package, depending on loss convergence criteria. Importantly, for all the models only one training is conducted, therefore not implementing the *sequential* estimation paradigm. This, of course, reduces the accuracy of the final posterior estimation [2] and should be corrected in future work.

2.4 Inference models

Neural posterior estimation This method, presented in [1], aims to directly approximate the true posterior $p(\theta|x)$ without the use of an explicit likelihood function (*likelihood-free inference).* Given a general conditional density estimator $q_\phi(\theta|x)$, one wants to find the optimal parametrization $\hat{\phi}$ so that $q_{\hat{\phi}}(\theta|x) \sim p(\theta|x)$. For an estimator consisting of a neural network, this optimization represents the network training, and its carried out minimizing the loss function

$$\mathcal{L}(\phi) = -\sum_{j=1}^{N} \log q_\phi(\theta_j|x_j) \tag{2}$$

Indeed, one can prove that minimizing (2) is equivalent to minimizing the expected KL divergence between $q_\phi(\theta|x)$ and $p(\theta|x)$

$$\mathcal{L}(\phi) = \mathbb{E}_{p(x)} \left[D_{KL} \left(p(\theta|x) \,||\, q_\phi(\theta|x) \right) \right] \tag{3}$$

so that for a trained network and an observation x_0

$$q_{\hat{\phi}} = \arg\min_{\phi} D_{KL} \left(p(\theta|x_0) \,||\, q_\phi(\theta|x_0) \right)$$

Importantly, this method is *amortized*, i.e. it provides a direct mapping $g_\phi : x \mapsto q_\phi(\theta|x)$ between the space of the observations and the space of conditional distributions of θ. Therefore, once the estimator is trained, no additional sampling is necessary to compute $q_{\hat{\phi}}(\theta|x_0)$ for any observation x_0.

Neural likelihood estimation NLE is algorithmically very similar to its counterpart, NPE. The main difference is that a neural density estimator is trained to approximate the likelihood instead of the posterior, by minimising the loss function

$$\mathcal{L}(\phi) = -\sum_{j=1}^{N} \log q_\phi(x_j|\theta_j) \tag{4}$$

Surprisingly, minimising the expected log likelihood is also equivalent to minimising the expected KL divergence. After training $q_\phi(x|\theta)$, we can evaluate it for any chosen observation x_0. However, to evaluate the posterior for a given observation, $p(\theta|x_0)$, either Markov Chain Monte Carlo (MCMC) sampling or Variational Inference (VI) is used for each x_0. While a detailed description of the two strategies is beyond the scope of this report, it is important to mention that MCMC, unlike VI, is asymptotically exact, but also computationally expensive. This aspect is relevant for the results of our NLE benchmarking experiments. We also remark that unlike NPE, NLE is not amortised and, together with the cost of sampling, it can easily become resource exhaustive.

3 Results

Neural posterior estimation After training the density estimator with different subsampling numerosities, we start by qualitatively investigating the parameter predictive power on different values of θ. We can observe that in order to obtain somewhat meaningful inference, at least 10000 points are necessary for the NPE model training (Fig. 1). This fact is exceptionally evident in the case where we need to infer values of D laying close to the boundaries of the prior, for which the proposal obtained with 10000 training points or less does not depart strongly from the initial uniform prior.

Following these observations, we try to quantitatively investigate the inference quality of the two most common Bayesian estimators (*Maximum a Posteriori* and *Expected a Posteriori*) for the models trained with 10000 and 50000 points respectively. We proceed by subdividing the θ space into a

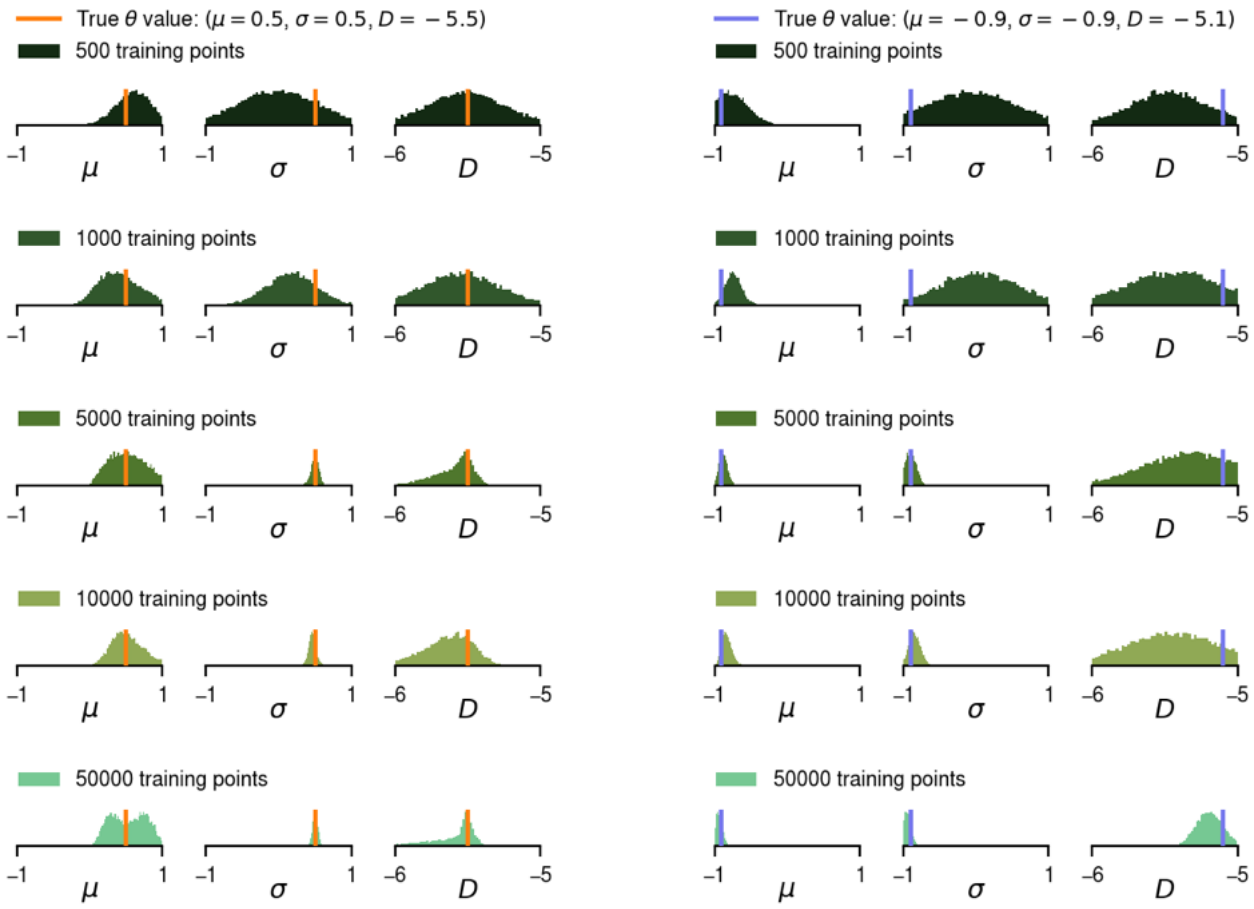


Figure 1: Posterior density estimates for two different values of true θ in function of the number of training points. By visual inspection, we can observe that meaningful inference occurs with at least 5000 training points, with the parameter D needing at least 50000 training points to be somewhat inferred in case of extremal value (right panel, lowermost row).

cubic grid of 27 elements, each containing a combination of the lowest, average, and highest values for the three parameters μ, σ and θ. For each cell, we then sample the proposal posterior with 10000 extractions, and compute the L^2 distance between the inferred θ and the optimal prediction, defined either as the maximum value (MAP) or the expected value (EAP) of the proposal[5].

Surprisingly, this evaluation does not find strong differences neither between training points nor between estimators (Fig. 2-7), but strongly suggests some kind of monotonic relation between the value of the diffusion coefficient D and the inference propensity of the corresponding θ parameter. Visual evaluation of the simulations, however, does not provide any insight about this fact but, on the contrary, would suggest that the σ parameter should have a much bigger impact on the inference power

[5]This method seems a bit of a freshman's dream. We can visualize it only because $\theta \in \mathbb{R}^3$, it is computationally prohibitive, and, most importantly, the L^2 could be not informative at all. It would be interesting, though, to study what metric one can use to obtain a meaningful distance in the parameter space. This question feels subtly connected to the parameter symmetries in the dynamical system itself - can this also be explored with SBI?

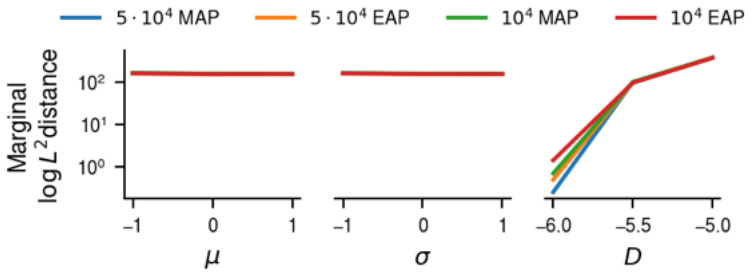
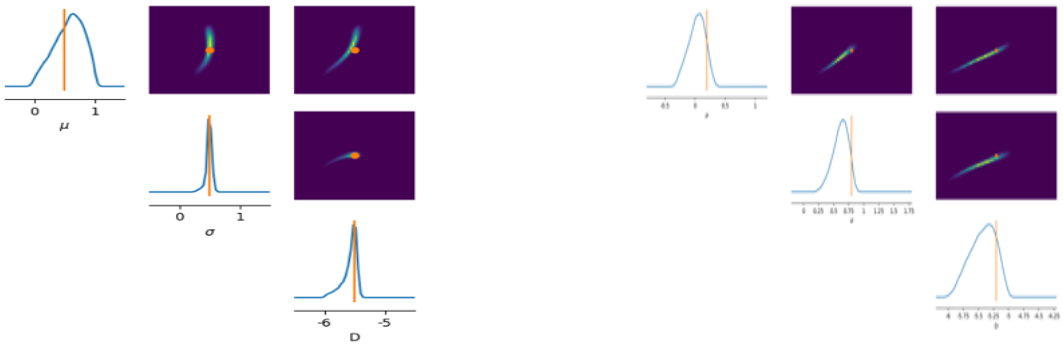


Figure 2: Cumulative ("marginal") L^2 distances from the estimate and the true θ in function of the θ components μ, σ, and D. Lower values of D seem to favour the inference of θ, although only under the used distance.



(a) θ centered

(b) θ extreme

Figure 3: Posterior predictive check at different parameter values

compared to D (Fig. 8). Further, more quantitative characterization is, therefore, necessary.

We further attempted to test inference quality using an out-of-the-box method provided by the SBI package (*Posterior Predictive Check*). Briefly, the posterior density estimator is first trained on a subsample of 10000 extractions from the ground truth joint based on prior estimates for effective training. We then simulate an observation x_0 using specified parameter value θ_0 and resample parameter estimates $theta_1$ from the posterior, setting x_0 as the default x. We then plot the distribution of resampled parameter estimates and see if they support the true parameter θ_0.

This evaluation reveals that the posterior predictive samples provide strong support around the true parameter values (Fig. 3a). However, the model does not perform as well when tested at values at the edges of parameter space (Fig. 3b).

Neural likelihood estimation Akin to NPE, NLE with high dimensional data ($100x100x3$) requires that we use summary statistics. However, given that NLE approximates the likelihood, it becomes necessary to compute the summary statistics outside the inference pipeline. It is unclear what summary statistics can be best employed for RGB images, particularly when the statistics differ from the well-researched natural images, as is the case for our RPS prey-predator model. Here, we test two hypotheses. First, we assume that the simulated images have homogeneous statistics, i.e. that

a patch has the same statistics as the original image. To this end, we sample from each simulation a $5x5x3$ image patch that we then flatten and use the resulting vector as input to the NLE pipeline. We thus reduce the number of dimensions of the simulated data from 30k to only 75 (Fig. 4). The alternative hypothesis is that the characteristic statistics of the images are linearly correlated and can be parsed using dimensionality reduction techniques (PCA). For each simulated observation we compute the number of principal components that explains most of the variance. We find that on average, 10 and 30 PCs are needed to explain 75% and respectively 95% of the variance in the data (Fig. 5).

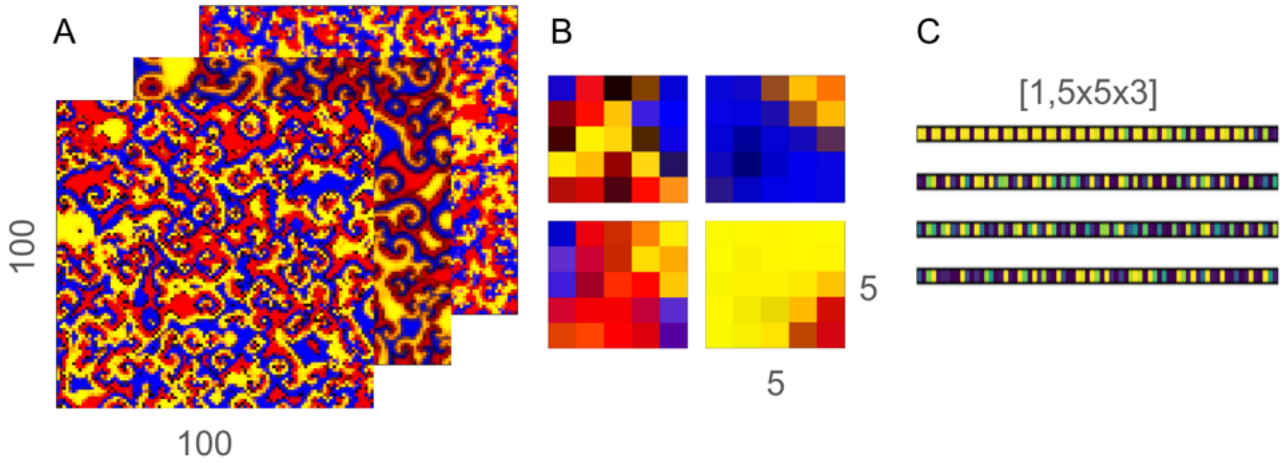


Figure 4: A) Simulated data with 2D rock-paper-scissors toy model. B) Low dimensional 5x5x3 patches sampled from the original simulated data. C) Flattened samples passed as input vector to the NLE pipeline.

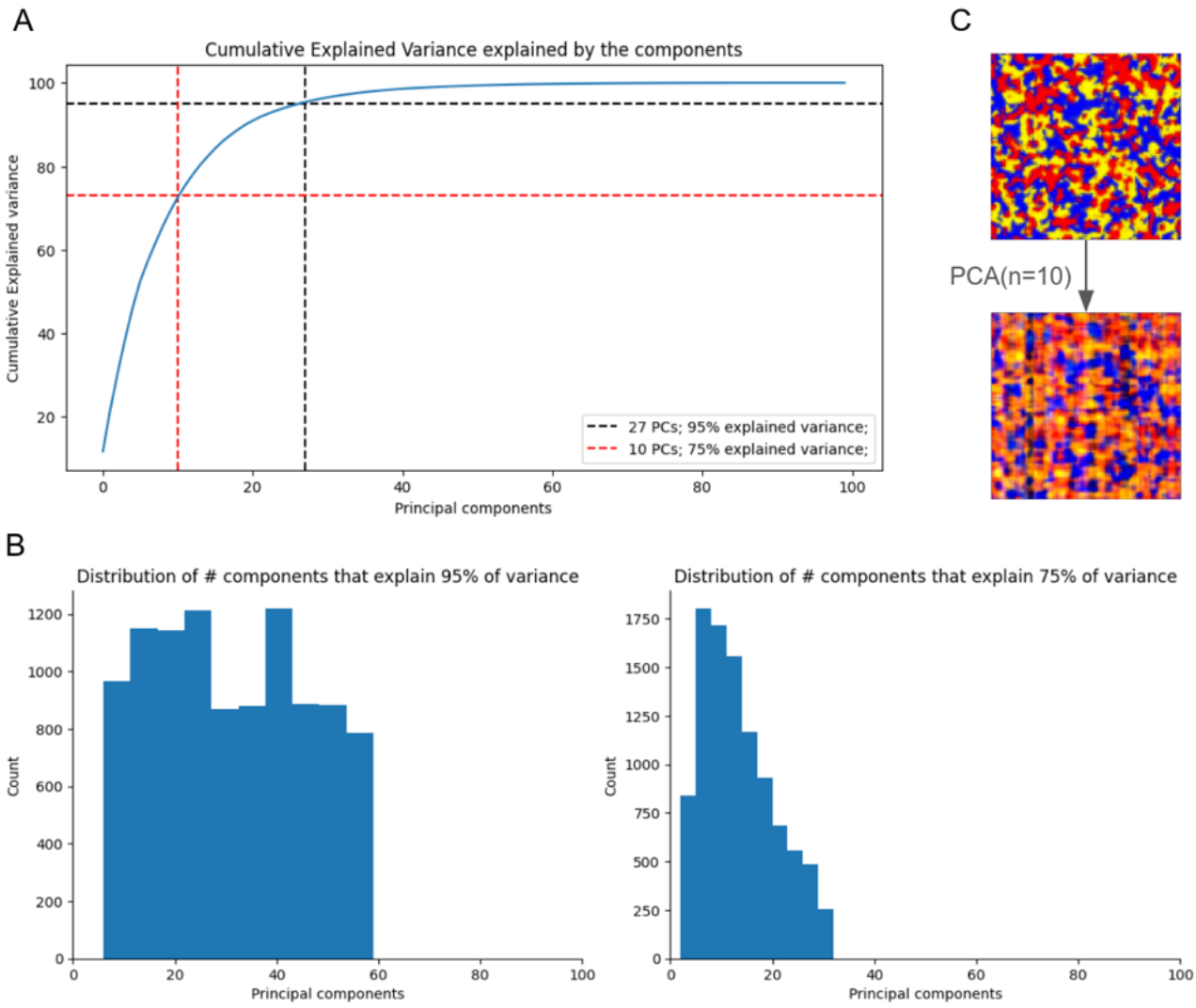


Figure 5: A) Cumulative explained variance as a function of the number of principal components used in reducing dimensionality of RPS simulated data. For one example, we observe that 10 (red) and 30 (black) PCs respectively are enough to explain 75% (red) and 95% (black) of the variance in the data. B) Left: histogram of the number of PCs explaining 95% of the variance across all simulated data. Right: histogram of the number of PCs explaining 75% of the variance across all simulated data. C) example of reconstruction of original data using 10 PCs.

The results of the NLE pipeline for evaluating the posterior at a given observation are unpromising. We plot the marginal posteriors for the three parameters of the model (μ, θ, D), along with their 2D pairwise distributions for each of the three summary statistics we investigated (Fig. 6). Using the RPS toy model means that we have access to the ground-truth parameters that generated the observation, x_0, and we can assess the quality of the inference. Results suggest that the statistics of the data are not homogeneous, given that inference performed worst with sampled patches from the original image (Fig. 6A). Using PCA to reduce the dimensionality of the data shows a modest improvement in the quality of the posterior. 10 PCs are however too few to capture enough variance

in the data (Fig. 6B), while 30 PCs explode the computational demand. As previously mentioned, NLE uses a sampling algorithm (MCMC or VI) to generate a posterior from the likelihood estimation. Using more than 10 PCs requires that we use VI which, despite being faster, is imprecise, leading to a degradation of the posterior despite using more PCs (Fig. 6C).

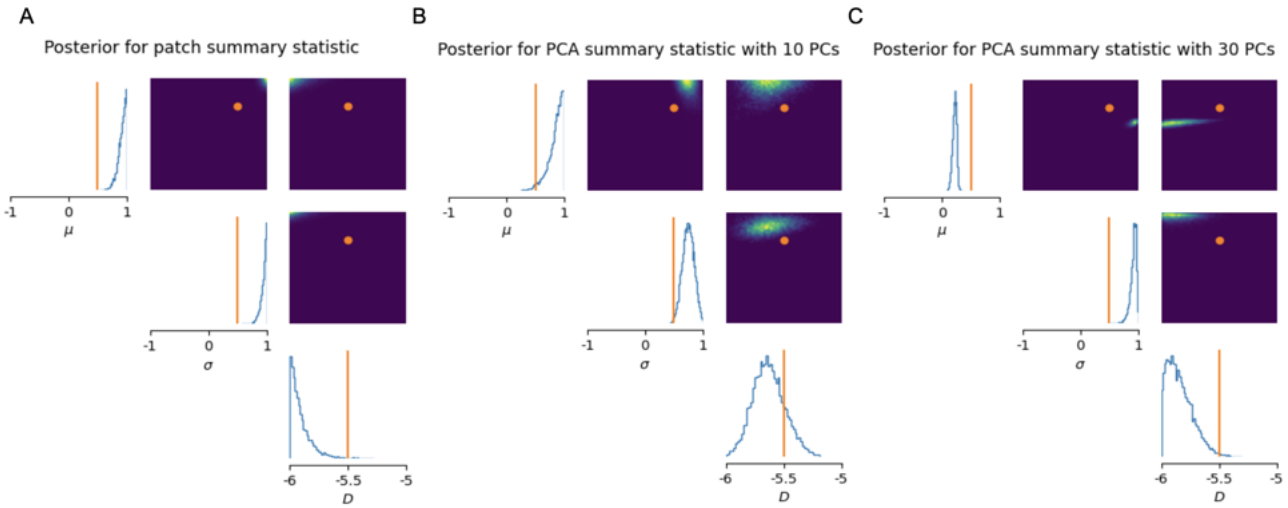


Figure 6: A) Posterior estimation using patch ($5x5x3$) samples from the simulated data. B, C) Posterior estimation using PCA to reduce dimensionality of the simulated data for: B) 10 PCs ($100x10x3$ input vector) and C) 30 PCs ($100x30x3$ input vector)

Overall, our results suggest that NLE is not a well-suited inference method for non-naturalistic images with unknown statistics. It is already known that likelihood estimation for high-dimensional data can be challenging. Finding low-dimensional summary statistics that capture most of the variance in the data is both time and computationally expensive, further pointing towards likelihood-free estimation as a more suitable inference approach.

4 Conclusions

Given that the problem at hand involves learning posterior parameter distributions given image data, it is clear that amortized methods like NPE handily outperform NLE both in terms of computational cost and accuracy of inferred parameter distributions. It is possible that NLE will prove more suitable for problem statements where the number of parameters exceeds the number of output elements, but this would require further investigation and a suitable problem statement.

5 Acknowledgements

We are very grateful to Maximilian Eggl and Laura Bernáez Timón for the seamless organisation of the workshop and for giving us the opportunity to be part of it. We are equally grateful to Pedro Gonçalves, Sara Vieira-Silva, Nastya Krouglova and Guy Moss for the countless hours they spent teaching, bug-fixing, guiding and inspiring us to do amazing science.

References

[1] David S. Greenberg, Marcel Nonnenmacher, and Jakob H. Macke. "Automatic Posterior Transformation for Likelihood-Free Inference". In: *International Conference on Machine Learning*. 2019. URL: https://api.semanticscholar.org/CorpusID:159041084.

[2] Jan-Matthis Lueckmann et al. "Benchmarking Simulation-Based Inference". In: *International Conference on Artificial Intelligence and Statistics*. 2021. URL: https://api.semanticscholar.org/CorpusID:231583170.

6 Supplementary

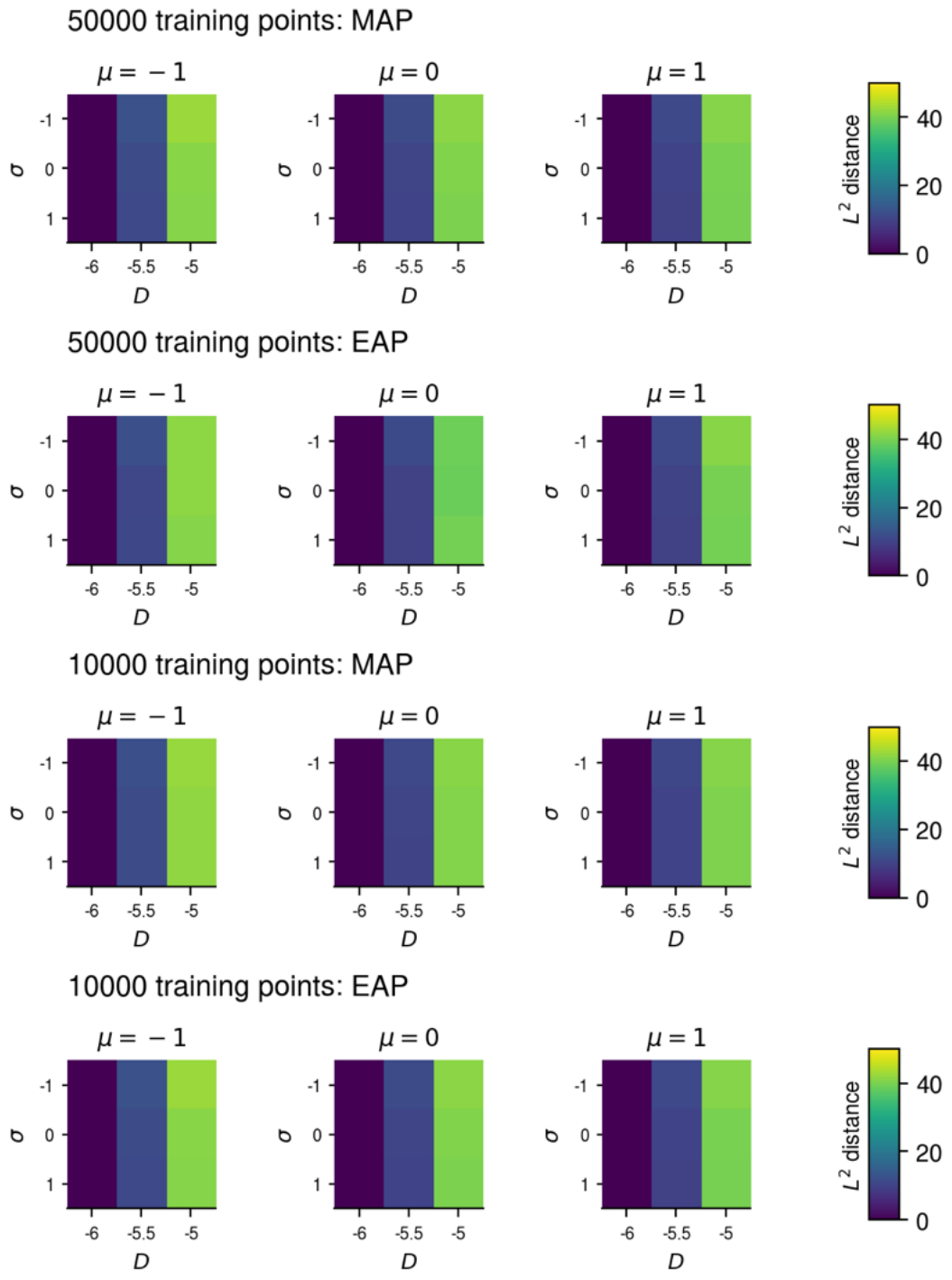


Figure 7: Grid evaluation of MAP and EAP estimators on the posteriors generated with 10^4 and $5 \cdot 10^4$ training points.

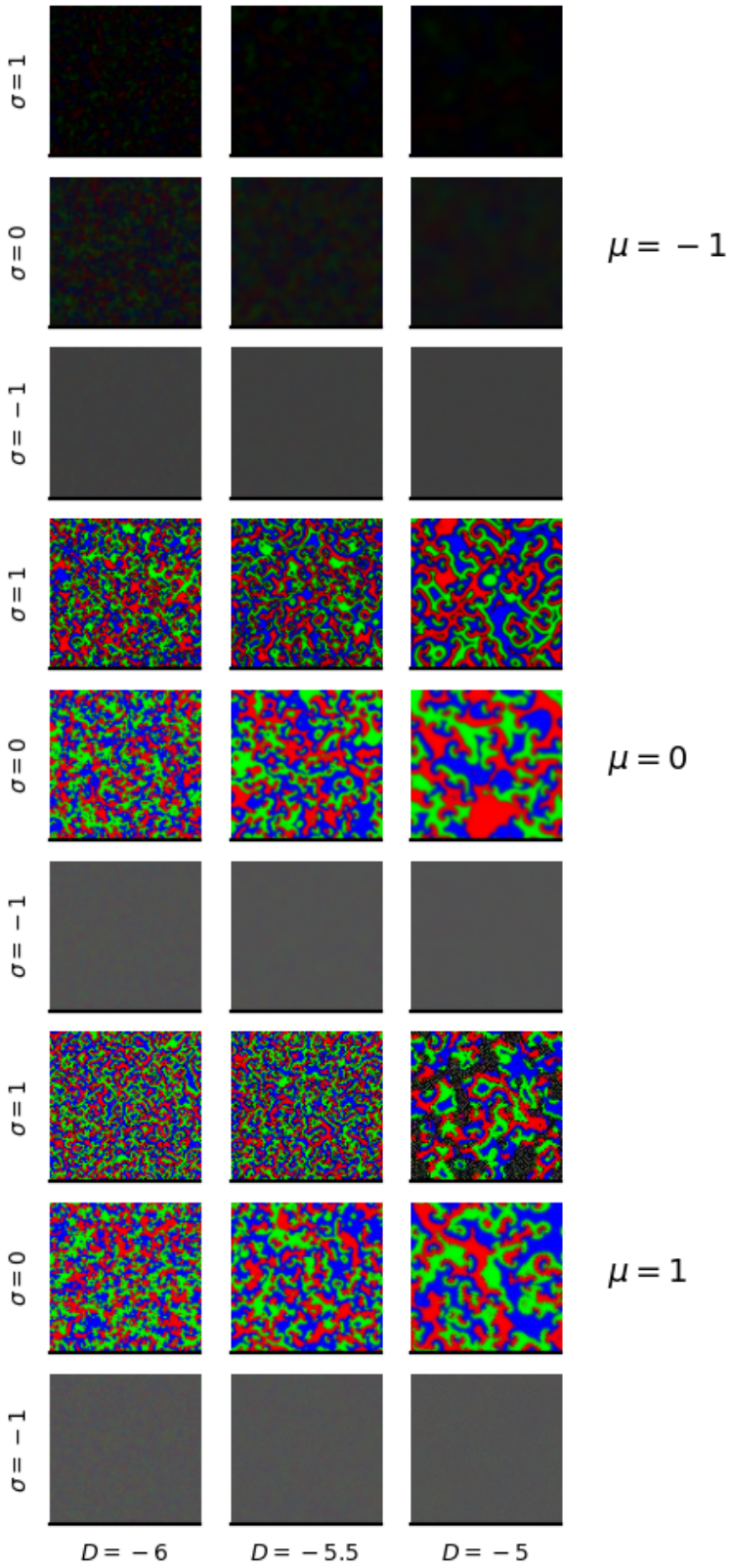


Figure 8: Simulations (x) corresponding to the explored estimator grid evaluation. Surprisingly, σ seems to impact the dynamics much more than D,[11] but the L^2 norm does not pick this up.

Simulation-based inference allows amortised learning for efficient model training

Olesia Dogonasheva[1], Albert Miguel López[2], Max Schelski [3], Yang Xu [4]

Abstract

Amortised inference is the process by which we train an inference network once using a large number of simulations. It allows subsequently to carry out rapid *amortized* parameter inference on new data using a single pass through the network. In this work, we used Simulation-Based Inference approach in order to train amortised logistic regression. We analysed its performance depending on the size of the dataset, the number of features, and the weights variance. We found that we needed 100,000 datasets for good amortized logistic regression performance using only two data features, while these were not enough datasets for more features. On synthetic data that is better separable through logistic regression, amortized logistic regression also was better at predicting the labels obtained by logistic regression. Finally, we allowed variable input data size through the use of gaussian summary statistics. In short, by approximating complex posterior distributions without explicit likelihood evaluations, SBI enables direct inference for new data points, as demonstrated in our application to logistic regression. This approach shows potential for improving model training processes and advancing machine learning research.

1 Introduction

Simulation-Based Inference (SBI) represents a paradigm shift in probabilistic modeling, enabling the approximation of complex posterior distributions through the generation of synthetic data [1]. This methodology avoids the need for explicit likelihood evaluations, making it particularly well-suited for scenarios where analytical solutions are intractable or computationally expensive. Moreover, SBI facilitates the training of amortized models, where a neural network is employed to learn a mapping from observed data to posterior distributions. This enables the amortized model to instantaneously infer posterior distributions for new data points, bypassing costly iterative inference procedures [2].

In this article, we explore the feasibility of leveraging SBI to train machine learning models in an amortized fashion, utilizing neural density estimators to compute posterior distributions efficiently for new data points. By using these techniques, one can overcome the limitations associated with model training, enhancing flexibility and performance. Our study focuses on applying SBI techniques specifically to logistic regression (LR), a fundamental machine-learning algorithm renowned for its simplicity and interpretability. Through empirical testing and analysis, we demonstrate the performance and scalability of this method, shedding light on its potential for enhancing model training processes.

2 Methods

The pipeline of the analysis is illustrated in Fig. 1. The process starts with the random sampling of feature vectors and weights from a standard normal distribution, denoted as $\mathcal{N}(0, I)$. Following this, a logistic regression (LR) model is simulated to generate corresponding labels based on these

[1]Institut Pasteur, Université Paris Cité, Hearing Institute, Paris, France
[2]Goethe University, Frankfurt, Germany
[3]Axon Growth and Regeneration Group, German Center for Neurodegenerative Diseases (DZNE), Bonn, Germany.
[4]Department of Connectomics, Max Planck Institute for Brain Research, Frankfurt, Germany

feature vectors. Subsequently, the Simulated Bayesian Inference (SBI) framework is applied, where the dataset undergoes reshaping and the LR model serves as the simulator. Within the SBI framework, the output is an estimator for the posterior distribution of LR weights given the dataset, encapsulating uncertainties in parameter estimation. The final step involves evaluating the efficacy of the selected optimal parameters using test dataset samples of varying sizes and sampling distributions ($\mathcal{N}(0, \sigma)$, $\sigma = 0.1, 1, 10, 100, 1000$).

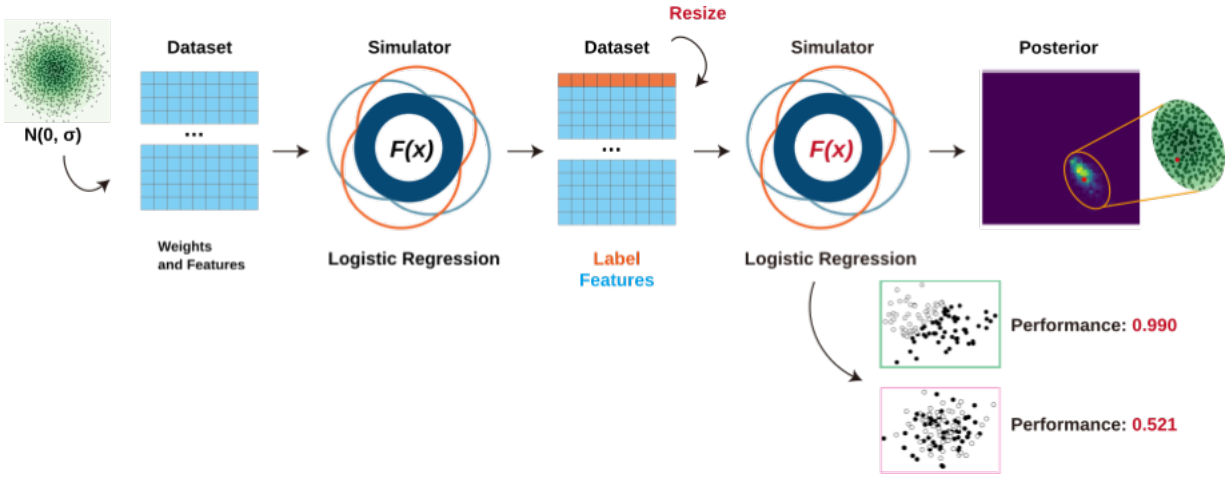


Figure 1: The analysis pipeline of amortized logistic regression with SBI. The first step is to sample feature vectors and weights from the normal distribution $\mathcal{N}(0, I)$. Then, simulate LR to obtain the corresponding labels. After that, run SBI framework, where the dataset is reshaped and simulator is LR. The SBI output is a posterior distribution estimator for LR weights given a dataset. The final step is the evaluation of the selected optimal parameters using test dataset samples of different sizes.

2.1 Dataset

The dataset consists of independently sampled feature vectors $\mathbf{X}$ and weights $\mathbf{W}$ from the multivariate normal distribution $\mathcal{N}(0, I)$, where I is the identity matrix. Given these, we calculated the probabilities of the logistic regression model, given by:

$$p = \sigma(\mathbf{W} \cdot \mathbf{X})$$

which we then used to simulate the labels y by sampling $y \sim Bernoulli(p)$. We generated synthetic data containing $100,000$ datasets DS (except for Fig. 3 where the number of datasets was varied), each containing 100 data points DP each with 2 features (except for Fig. 6 and Fig. 7 where the number of features was varied).

To speed up data generation for a large number of datasets, we wrote a GPU kernel in Python using the package numba (numba.cuda.jit) to generate label probabilities in parallel for all datasets on a GPU.

2.2 Simulation-based inference

For simulation-based inference, we used the sequential neural posterior estimator of the Python package sbi. For data embedding, we used a fully connected embedding network with 2 layers, 500 hidden

units, 20 output units, and input dimensions $IN = (F+1) \cdot DP$, where F is a number of features, DP is a number of data points. A mixed-density network was used as a density estimator. The dataset had a shape of $[DP, IN]$ with parameters equaling the LR weights of each dataset with shape $[DP, F]$ and a batch size of 10000 until convergence, as determined by the sbi package.

2.3 Evaluation

To analyse the performance of SBI, we generated 1000 datasets. Then, we computed LR labels using scikit-learn for each dataset separately.

To calculate amortised logistic regression (aLR) labels, we sampled 1000 weight sets from the trained weight posterior distribution. To measure the accuracy of true labels, we calculated the number of LR or aLR labels matching the true labels and normalized it with respect to the total number of data points. To measure the accuracy of LR labels, the number of aLR labels matching LR labels was normalized with respect to the total number of data points.

2.4 Software

All code was programmed in Python 3.10 using the following packages: pytorch 2.2.2, sbi 0.22.0, numpy 1.26.4, numba 0.59.1, pandas 2.2.1, matplotlib 3.8.4, seaborn 0.13.2, scikit-learn 1.4.1.post1.

3 Results

3.1 Parameter Space Representation

To explore the representation of simulator parameters, all parameters from multiple training sessions were rated in Figure 2, left. Each row within the dataset denotes a distinct parameter, with the total count denoted by F, contingent upon the number of data features incorporated.

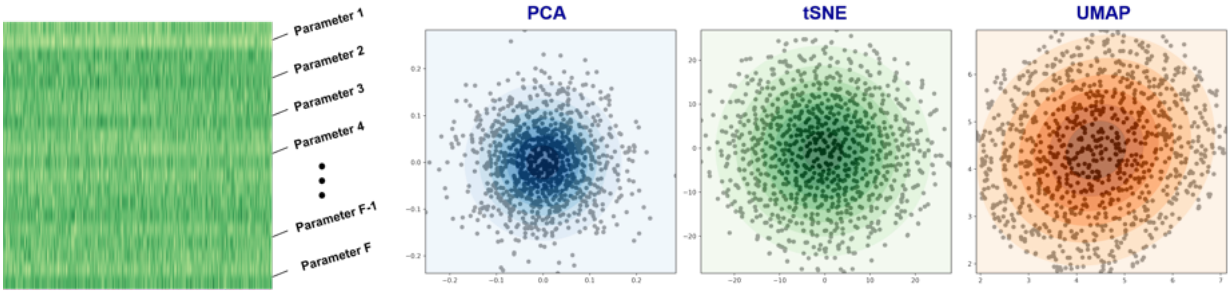


Figure 2: Left: A high-dimensional parameter space graph comprising all parameters. Right: Parameter feature space under Gaussian fitting.

We applied carious data dimensionality reduction methodologies, including Principal Component Analysis (PCA), Uniform Manifold Approximation and Projection (UMAP), and t-distributed Stochastic Neighbor Embedding (tSNE), to condense the high-dimensional parameter space into a two-dimensional visualization (Fig. 2, right).

3.2 Performance

To test the performance of aLR, we trained a sequential neural posterior estimator (SNPE) for aLR weights. First, we tested the dependency of aLR performance on the number of datasets. We evaluated the accuracy of aLR to predict LR labels (aLR accuracy). While 10^2 or 10^3 datasets led to a close-to-chance (50%) accuracy, 10^4 and 10^5 showed an accuracy of $\sim 80\%$ and $\sim 90\%$ respectively (Fig. 3). For further analysis, we used 10^5 datasets.

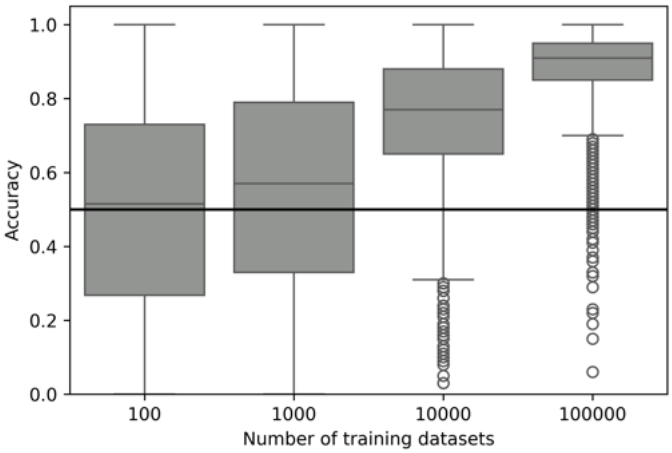


Figure 3: Accuracy of aLR predicting LR labels depending on the number of datasets.

Next, we analysed the dependence of this performance on the number of features. Here, we first evaluated the accuracy of LR and aLR on unseen synthetic data compared to true labels (true accuracy, Fig. 4, left). As expected, for LR, true accuracy increased with an increasing number of features due to the better separability of data in a space of higher dimensions. However, for aLR (posterior, Fig. 4, left), the true accuracy decreased monotonically with an increasing number of features, reaching a chance level of 16 features (Fig. 4), left.

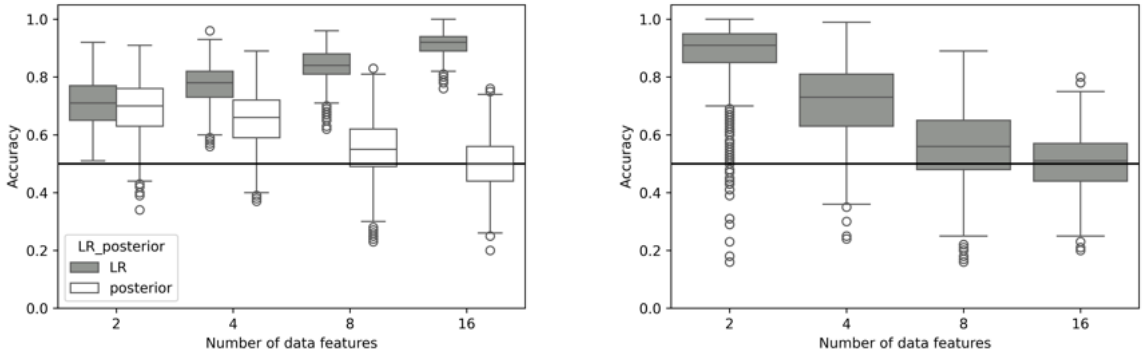


Figure 4: Left: True accuracy of LR and aLR depending on the number of data features. Right: Accuracy of aLR predicting LR labels depending on the number of data features.

Indeed, aLR accuracy similarly decreased to the chance level at 16 features (Fig. 4, right).

Since 10^5 datasets were already needed for good accuracy with 2 features (Fig.4, left), probably many more datasets would be needed for good aLR performance with more data features. Thus, we next checked the influence on aLR accuracy of how well LR can predict true labels (true accuracy of LR), i.e. how separable the data is by LR. We generated synthetic data with varying degrees of separability by increasing the variance of weights to generate the data (Fig. 5).

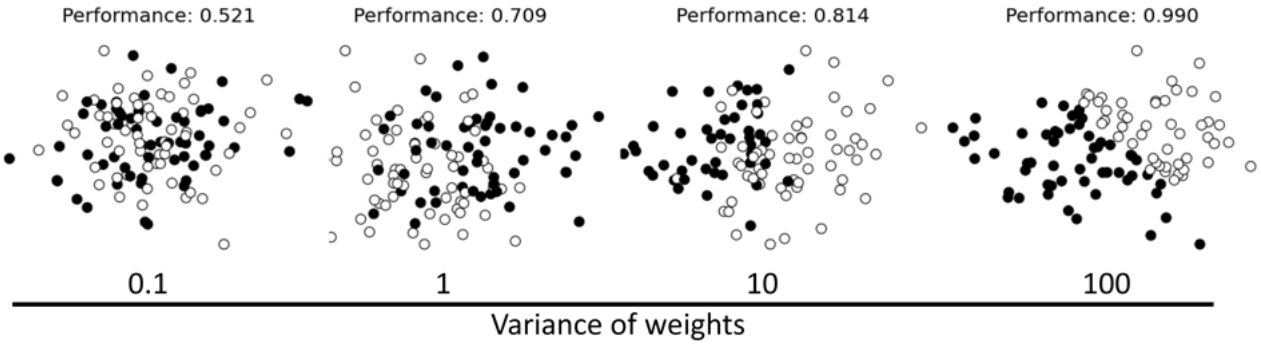


Figure 5: Changing separability of data. Performance indicates accuracy of predicting true labels by LR. Variance of weights indicates variance of multivariate gaussian distribution of weights used to generate synthetic data.

As expected, the true accuracy of both LR and aLR increased with increased separability (Fig. 6).

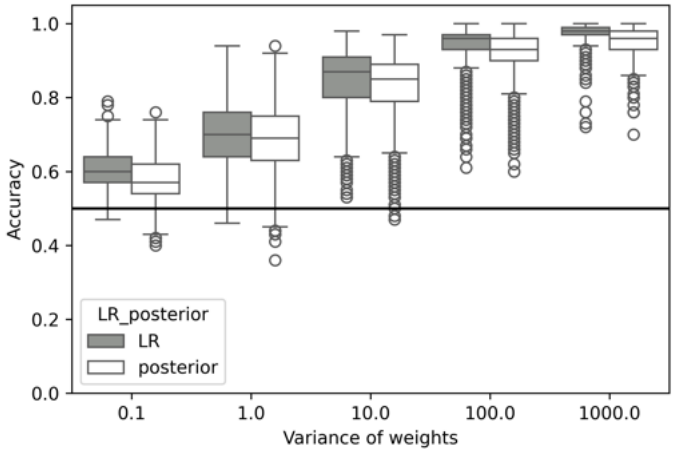


Figure 6: True accuracy of LR and aLR depending on the separability of data. A higher variance of weights indicates data is more easily separable by LR.

Interestingly, aLR accuracy increased monotonically with increased data separability up to $\sim 96\%$ (Fig. 7).

This could either indicate that the posterior is easier to learn for more easily separable data or that similar errors in predicting the weights of logistic regression lead to a reduction in accuracy with more easily separable data.

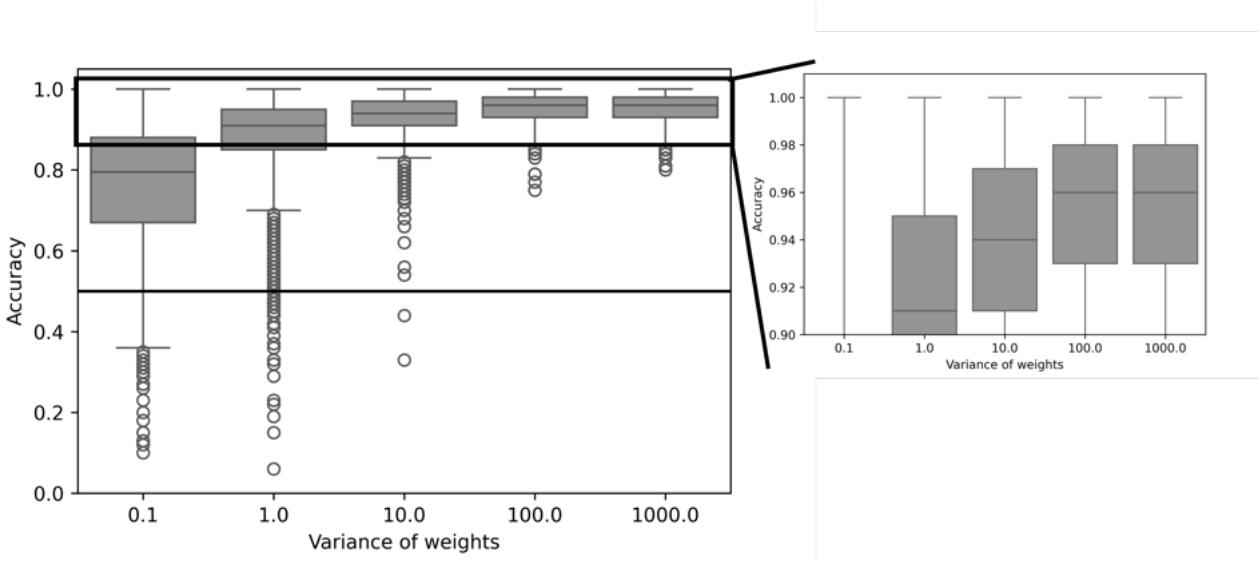


Figure 7: Accuracy of aLR predicting LR labels depending on the separability of data. A higher variance of weights indicates data is more easily separable by LR.

Finally, we trained SNP as a mixture density network model using a fully connected network (2 layers, 1000 hidden states, output dimension was 5) and permutation invariant embedding with summation as an aggregation function (2 layers, 100 hidden states, aggregation dimensionality was 1, output dimension was 20). The training batch size was 10000. Figure 8 shows the histogram of accuracy for the classification task when weights were found by standard LR, amortised LR, and were chosen randomly. The mean accuracy for aLR is 0.977, std= 0.021.

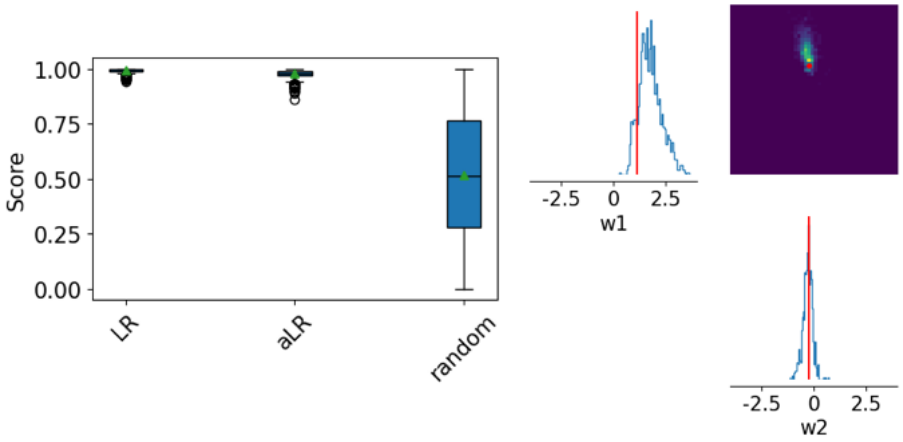


Figure 8: Left: classification accuracy using LR, aLR and random weights. Right: resulting posterior distribution of LR weights.

3.3 Variable Dataset Size

One issue with previous approaches is that they require the number of samples in each DS to be equal. This is of course not a realistic condition, as real datasets may have any possible size. One strategy to accomodate variable DP number is to transform the data using a function with a set output dimensionality. Following the idea of a "summary statistic", we thought to model each class in a dataset by a gaussian distribution, which is defined only by its mean and covariance. The intuition is to reduce each class to the statistics of the gaussian distribution that fits the points best.

If the original dimensionality is F, each gaussian model will be defined by its mean vector μ of size F, and its covariance matrix C of size $[F, F]$. The new input to the neural network will be these statistics concatenated in a vector:

$$\left[\mu_1^{(1)}, ..., \mu_F^{(1)}, C_{11}^{(1)}, ..., C_{1F}^{(1)}, C_{21}^{(1)}, ...C_{FF}^{(1)}, \mu_1^{(2)}, ..., \mu_F^{(2)}, C_{11}^{(2)}, ..., C_{FF}^{(2)}\right]$$

Where $\mu_i^{(c)}$ is the mean for the dimension $i = 1, ..., F$ for class $c = 0, 1$, and $C_{jk}^{(c)}$ the covariance for dimensions j and k. For two classes, the resulting vector will be of size $2 \cdot (F + F^2)$, which is independent of datapoints DP.

We evaluated this method using the tools described previously, with the network described in section 2.2. The only differences being that the input datasets were assigned each a different randomized DP size, chosen as a random number between 10 and 500 with uniform probability, and that the input to the network is the gaussian summary described above.

Results are shown in figure 9, for both high and low class separability. The amortised networks tracks the original logistic regression almost perfectly in both cases, showing that summarizing a dataset using gaussian models does not diminish model performance. Moreover, this method is much faster in training than previous ones, as input size has been lowered considerably. For example, for F=2 and DP=100, our input dimension was $IN = (F+1) \cdot DP = 300$, while using a gaussian summary statistic it reduces to $IN = 2 \cdot (F + F^2) = 12$.

As for possible disadvantages, we need to mention at least two. This summary statistic takes some extra steps to compute, which means the amortised computing time might be higher (we may expect this for larger amounts of larger datasets). And most importantly, we have no guarantee that this method will generalize to other input datasets or regression models. Our data is generated using gaussian distributions, so it makes sense that summarizing them by their gaussian statistics gives good results. But it's possible that this method will give worse results when the input classes are clearly non-gaussian (e.g. half-moon datasets). Also, logistic regression is a fairly simple algorithm, intuitively it simply separates space in two classes with a simple plane. More complex classification algorithms with complicated class boundaries might use details of the dataset not covered in the gaussian statistics for classification, meaning that this method would eliminate too much information and fail. These and other shortcomings could be examined in future work.

4 Conclusions

In conclusion, Simulation-Based Inference (SBI) coupled with amortized learning techniques presents a promising avenue for enhancing the efficiency and scalability of machine learning model training. By approximating complex posterior distributions through the generation of synthetic data and leveraging neural density estimators to compute posterior distributions efficiently, SBI enables the training of models in an amortized fashion. This approach sidesteps the need for costly iterative inference

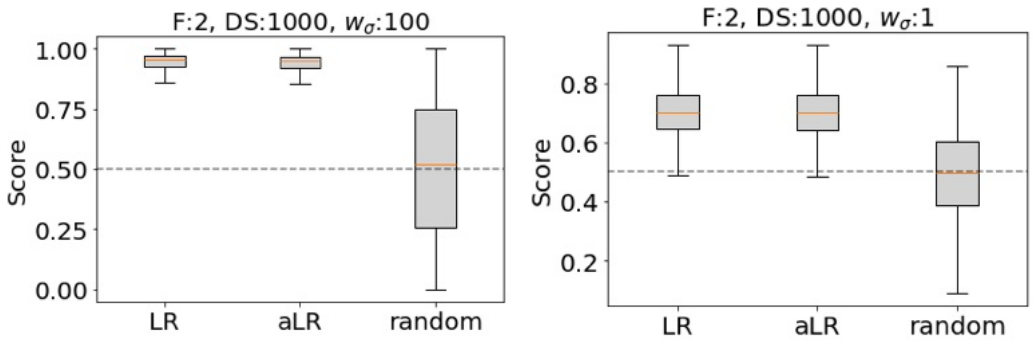


Figure 9: Classification accuracy using LR, aLR and random weights for inputs with variable DP size, summarized using gaussian modeling. Results shown for both high (left) and low (right) class separability.

procedures, allowing for instantaneous inference of posterior distributions for new data points. Our exploration of applying SBI techniques to logistic regression demonstrates the feasibility and potential of this approach, showcasing its ability to overcome limitations associated with traditional model training methods. Through empirical testing and analysis, we have illustrated the performance and scalability benefits of this method, highlighting its capability to improve model training processes and open new avenues for research in machine learning. As machine learning continues to evolve, the integration of SBI and amortized learning techniques holds promise for further advancing the field, enabling more efficient and effective model training across a wide range of applications.

5 Acknowledgements

We are grateful to the SBI workshop organizers, Maximilian Eggl and Laura Bernaez Timon. Special thanks are due to Pedro Gonçalves for his exceptional lectures, and to Guy Moss and Nastya Krouglova for their unwavering assistance throughout.

6 References

References

[1] Cranmer, Kyle, Johann Brehmer, and Gilles Louppe. "The frontier of simulation-based inference." Proceedings of the National Academy of Sciences 117.48 (2020): 30055-30062.

[2] Marino, J., Yue, Y., & Mandt, S. (2018, July). Iterative amortized inference. International Conference on Machine Learning (pp. 3403-3412). PMLR.

Summary statistics for simulation-based inference

Surbhit Wagle[1], Douglas Feitosa Tomé[2], Alpcan Önür[3], and Rune Höper[4]

[1] Institute of Experimental Epileptology and Cognition Resaerch, University Clinic Bonn, Germany
[2] Institute of Science and Technology Austria, Klosterneuburg, Austria
[3] Leibniz Institute for New Materials, Saarbrücken, Germany
[4] Institute for Theoretical Biology, Humboldt University Berlin, Berlin, Germany

Abstract. Simulation-based inference (SBI) is a statistical approach used to estimate parameter distributions by leveraging simulations instead of relying exclusively on theoretical distributions or analytical derivations. Summary statistics are used to reduce the dimensionality of the simulated data used as input in SBI. Different methods to achieve this reduction exist, either manually curated or by automated techniques. One of the key open questions in the SBI field is whether we should learn summary statistics (SuSt) and neural density estimators in parallel (i.e., coupled approach) or separately (i.e., decoupled approach) when considering the properties of the inferred posteriors and their associated computational costs. In this work, we made an initial attempt to address this question by using different methods to compute SuSt and then comparing the properties of the inferred posterior distributions. We used the Hodgkin-Huxley model of a single neuron as a case study and computed SuSt using i) expert-defined statistics, ii) variance-preserving dimensionality reduction techniques, namely principal component analysis (PCA), iii) autoencoders, and iv) embedding networks in the SBI loop. Our preliminary results showed that each approach to compute SuSt yielded reasonable posteriors. However, expert-defined statistics and PCA but not embedding networks generated outliers in Euclidean distance space. Our results also pointed to differences in computational cost between coupled (i.e., embedding networks) and decoupled (i.e., expert-defined statistics, PCA, and autoencoders) approaches to compute SuSt, reflecting relative differences in the cost of adding an embedding network and the cost of algorithms used to generate SuSt offline.

Keywords: Simulation-based inference · Summary statistics · Expert-defined statistics · Principal component analysis · Autoencoder · Embedding network · Hodgkin-Huxley model

Introduction

Simulation-based inference (SBI) is a statistical approach that uses computer simulations from a prior parameter space to generate inferences over real world data without relying on likelihood functions like classical Bayesian inference. While a key component of SBI is the use of an appropriate simulator, another equally important one is the use of appropriate input data. Notably, simulators have become more refined and complex, leading to the production of high dimensional datasets to capture intricate feature spaces in real world phenomena. This high dimensionality can make training neural density estimators (NDEs) in SBI significantly more challenging. To address this, it has been proposed to replace the raw simulator output data used in SBI with summary statistics (SuSt) that capture key features of the original simulator output but with reduced dimensionality [1]. Importantly, SuSt can be computed separately from the training of NDEs (i.e., decoupled approaches) (Fig. 1) or they can be learned jointly with NDEs in parallel (i.e., coupled approaches) (Fig. 2). However, it is unclear whether the use of decoupled or coupled approaches to compute SuSt is more advisable when considering the properties of the inferred posteriors and the associated computational costs.

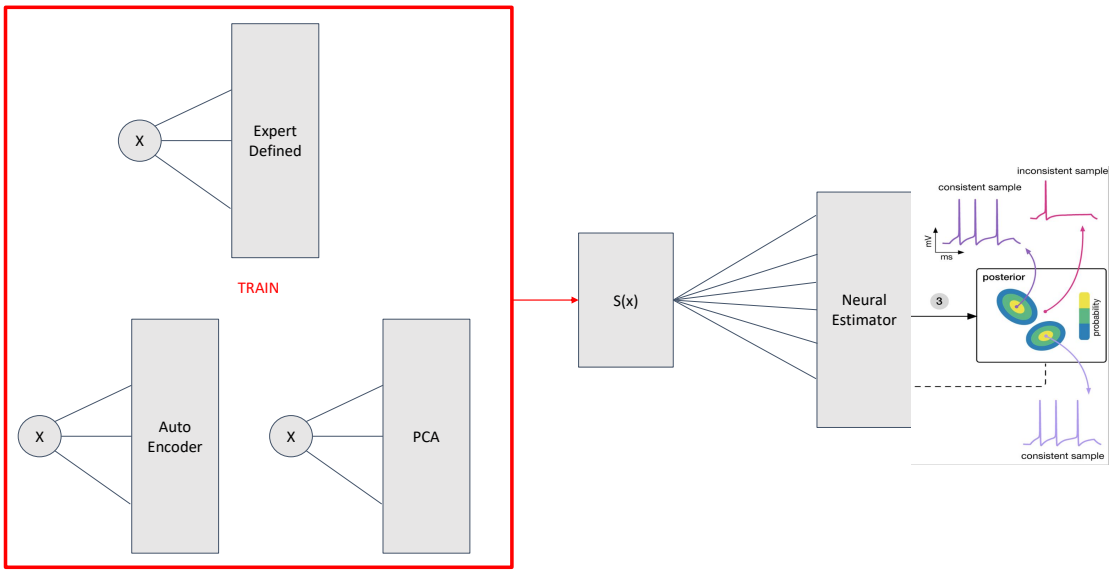


Fig. 1: **Computing summary statistics outside the SBI loop.** First, summary statistics are computed offline using either expert-defined statics, PCA, or an autoencoder. Then, the computed low-dimensional representations are used as input in the SBI loop.

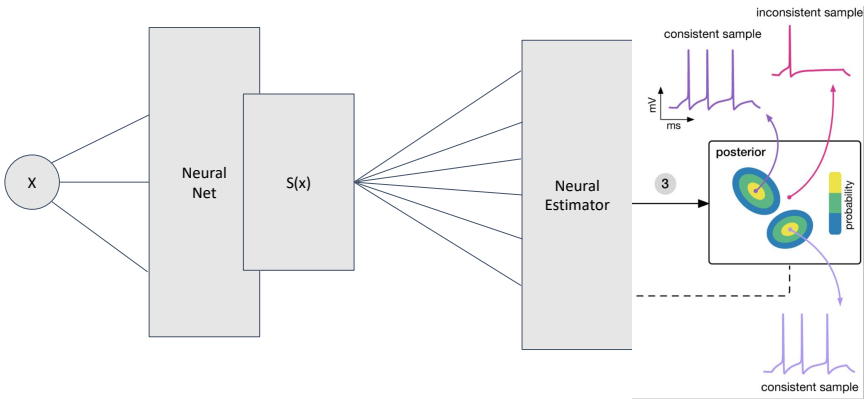


Fig. 2: **Learning summary statistics in the SBI loop.** High dimensional input data is directly used as input in the SBI loop. The dimensionality of the input data is reduced using an embedded neural network. The neural density estimator as well as the embedded neural network are trained at the same time.

Here, we addressed this question in the case of the Hodgkin-Huxley model (Fig. 3). This is a well-known mathematical model of single neurons originally proposed by Alan Hodgkin and Andrew Huxley in 1952 that consists of nonlinear differential equations. Briefly, this model describes the generation of action potentials in excitable cells. Specifically, the lipid membrane is described as a capacitance (C_m), voltage gated ion channels (g_n), and leak channels (g_L) as electrical conductances connected to voltage sources (E_n and E_L). In addition, ion pumps act as current sources (I_p) and the membrane potential is described by V_m. We employed SBI to infer posterior distributions of the parameters of the Hodgkin-Huxley model when using different decoupled and coupled approaches to compute SuSt. We found that each approach to compute SuSt yielded reasonable posteriors and that decoupled but not coupled approaches led to outliers in Euclidean distance space. Furthermore, we identified differences in computational costs across these approaches.

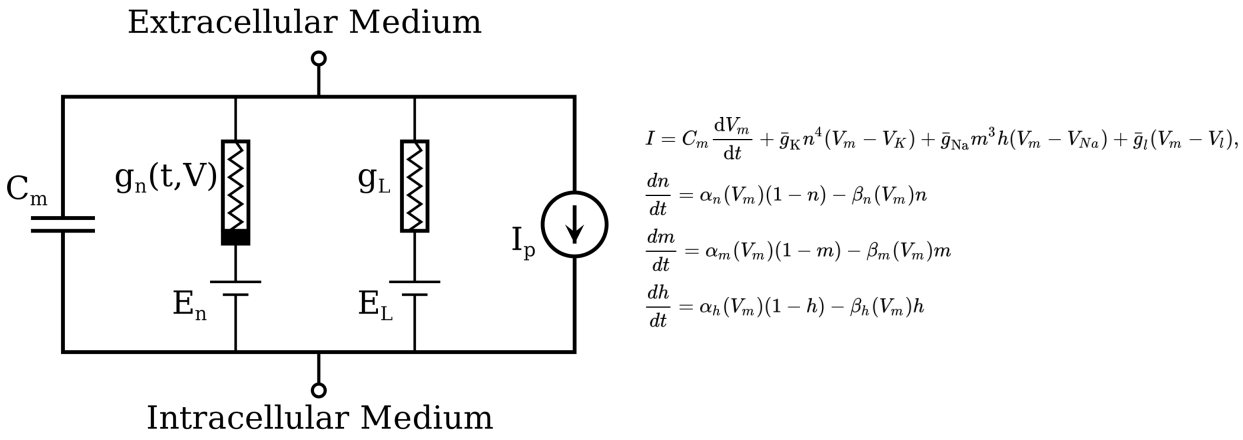


Fig. 3: **Schematic overview of the Hodgkin-Huxley model.**

Methods

We employed SBI to infer the posterior distribution of eight parameters of the Hodgkin-Huxley model when using different approaches to compute SuSt. Specifically, we used a dataset of 100,000 simulations of the Hodgkin-Huxley model where the eight parameters of interest were drawn from uniform prior distributions. We then either i) computed SuSt offline and used them as input to train a NDE (i.e., decoupled approaches: expert-defined statistics, PCA, and autoencoders) (Fig. 1) or ii) trained an embedding neural network and a NDE in parallel (i.e., coupled approach) (Fig. 2).

To compare the performance of the inferred posteriors when using different approaches to compute SuSt, we measured the Euclidean distance between a set of six expert-defined statics of an example trace and samples from each inferred posterior conditioned on the same example trace. We also measured the time to compute SuSt offline in the case of decoupled approaches as well as the time to train NDEs for both decoupled and coupled approaches to

evaluate the computational cost of each approach.

For the coupled approach to compute SuSt, our embedding neural network consisted of a fully connected network with 30,000 input units, a hidden layer of 50 units, and an output layer of six units (to match the set of six expert-defined statistics). For the decoupled approaches to compute SuSt, we first used a set of six expert-defined statistics. Second, we also used PCA and an autoencoder as alternative dimensionality reduction methods to compute SuSt. Briefly, PCA computes the covariance matrix to elucidate how features in the input vary together. By calculating the eigenvalues and eigenvectors of this matrix, PCA identifies the principal components, which capture the most variance in the data. Finally, PCA projects the original data onto the subspace defined by these principal components, providing a lower-dimensional representation that retains the most significant information. We chose to retain the first six principal components to match the number of expert-defined statistics. Similarly, an autoencoder is an unsupervised deep learning method that can be used to represent an input in a lower-dimensional latent space. This lower-dimensional representation can then be used as SuSt for SBI. An autoencoder is comprised of an encoder network, compressing input data into a lower-dimensional representation (latent space), and a decoder network, reconstructing the original input from this latent representation. Here, we used one *Conv1D* layer followed by one *MaxPool1d* layer and one *Linear* layer in our encoder. We also used a *ReLU* activation function after the max-pooling and linear layers. Our decoder consisted of a *Linear* layer followed by two *ConvTranspose1d* layers and a final *Linear* output layer (Fig. 5A). We used mean squared error as our loss function for training the autoencoder and used 1000 samples for calculating the training loss. We also set the dimensionality of the latent representation of our autoencoder to six to match that of the set of expert-defined statistics.

Results

We performed PCA using 100,000 simulations of the Hodgkin-Huxley model and observed that the first three principal components could explain $> 70\%$ of the variance in our input traces (Fig. 4A). Therefore, we used these three components to compute mutual information (MI) as a measure of how much each of the principal components explained the input parameters (Fig. 4B-D). We found that only the first principal component showed a high MI score ($\mathrm{MI} > 1$) with the parameter $\mathrm{E_{leak}}$, hinting that PCA may not yield an effective approach to compute SuSt for SBI.

Our autoencoder, named *TimeSeriesAutoEncoder* (Fig. 5A), experienced an initial reduction in the mean squared error (MSE) training loss but this was followed by a plateau without much further improvement, leaving the loss at a high value (> 200) even after 60 training epochs (Fig. 5B). This means that the the training procedure was not able to arrive at a well optimized autoencoder. For instance, we could see that our autoencoder was able to approximate the sub-threshold traces quite well (Fig. 5C) but could not match the spike count or spike timing of the input data (Fig. 5D.)

We measured the Euclidean distance between the set of expert-defined statistics of an example trace and that of samples drawn from either the prior uniform distributions (Fig. 6a) or the posterior distributions inferred using expert-defined statistics (Fig. 6b), PCA (Fig. 6c), and an embedding neural network (Fig. 6d). Notably, each of the approaches to compute SuSt (i.e., expert-defined statistics, PCA, and embedding network) substantially decreased the Euclidean distance between the example trace and samples drawn from the posterior distributions when compared to samples drawn from the prior distributions (i.e., baseline). Surprisingly, decoupled (i.e., expert-defined statistics and PCA) but not coupled (i.e., embedding network) approaches to compute SuSt led to outliers in Euclidean distance space. However, expert-defined statistics yielded the smallest median Euclidean distance (i.e., median Euclidean distance ≈ 9, Table 1), followed by the embedding network (median Euclidean distance ≈ 89, Table 1) and PCA (median Euclidean distance ≈ 112, Table 1). These results suggest that learning SuSt and NDEs in parallel reduces the rate of or even prevents outlier samples in the posterior distribution at the cost of a higher median Euclidean distance relative to computing expert-defined statistics offline.

Lastly, we found that the training time was the longest when using an embedding neural network compared to using expert-defined statistics or PCA (Table 1). This was expected given the high dimensionality of the input data and the consequent increased computational cost of training both the embedding network and the NDE in a coupled

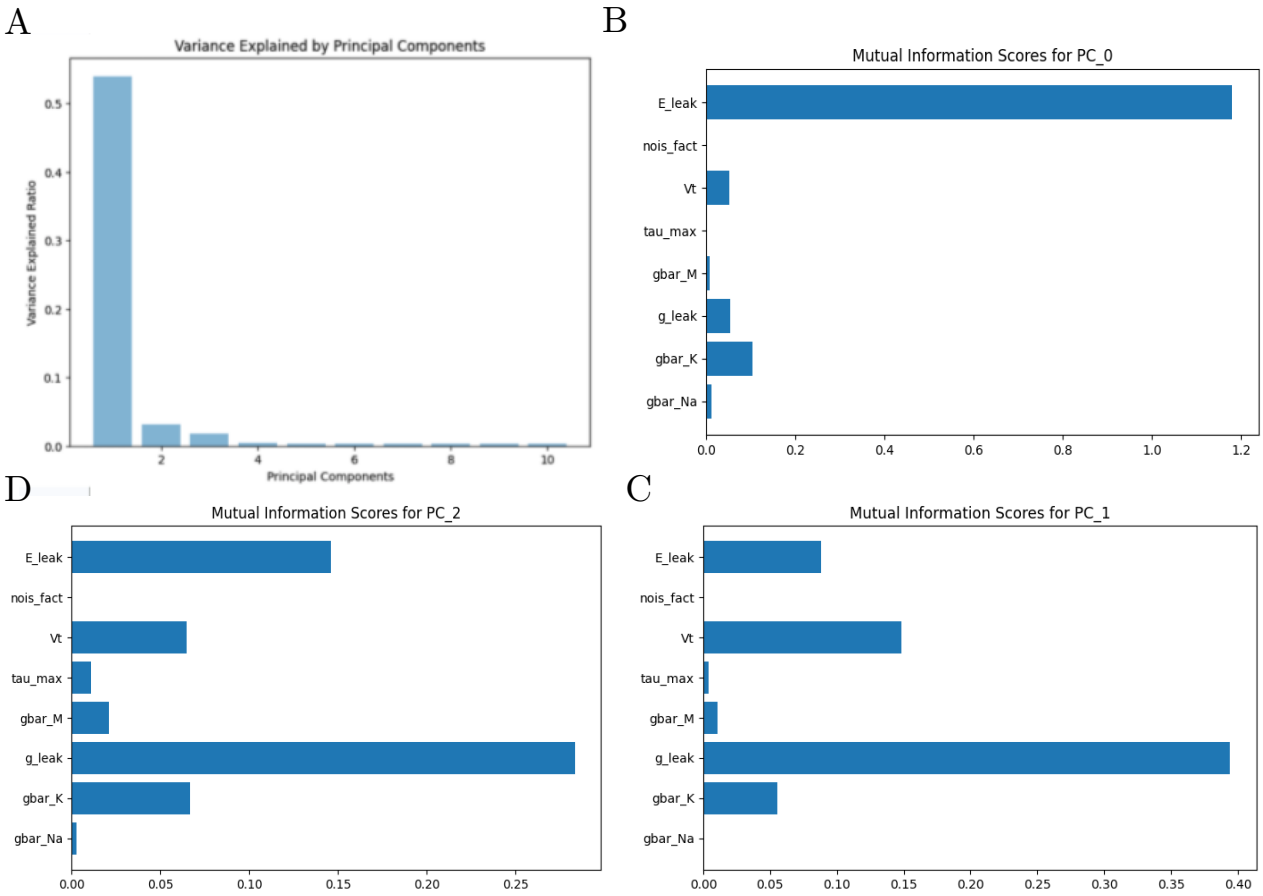


Fig. 4: **Principal component analysis as a method to compute summary statistics**.
A The first three components of the PCA could explain $>$ 70% of the variance in our input and, hence, can be used as SuSt. **B-D** Mutual information carried by the first three principal components for each of the parameters in the Hodgkin-Huxley simulations.

manner. However, the increase in training time when using an embedding network could be alleviated by using a GPU.

Model	Training time	Euc. distance median	Euc. distance std
Prior	-	489.77	169.80
Expert	28 min + statistics	8.77	123,725.82
PCA	18 min + 20 min statistics	112.08	86,659.96
Embedding Network	68 min (43 min on GPU)	88.83	63.33

Table 1: **Training time and Euclidean distance.**

Discussion

Both decoupled and coupled approaches to compute SuSt produced reasonable posteriors, comparable in performance to the curated statistics defined by experts in the research field. This suggests that effective SuSt for complex datasets can be computed automatically without the need for labour-intensive, human-defined SuSt. While there are qualitative differences, with the expert-defined statistics seemingly yielding the best results in terms of Euclidean distances

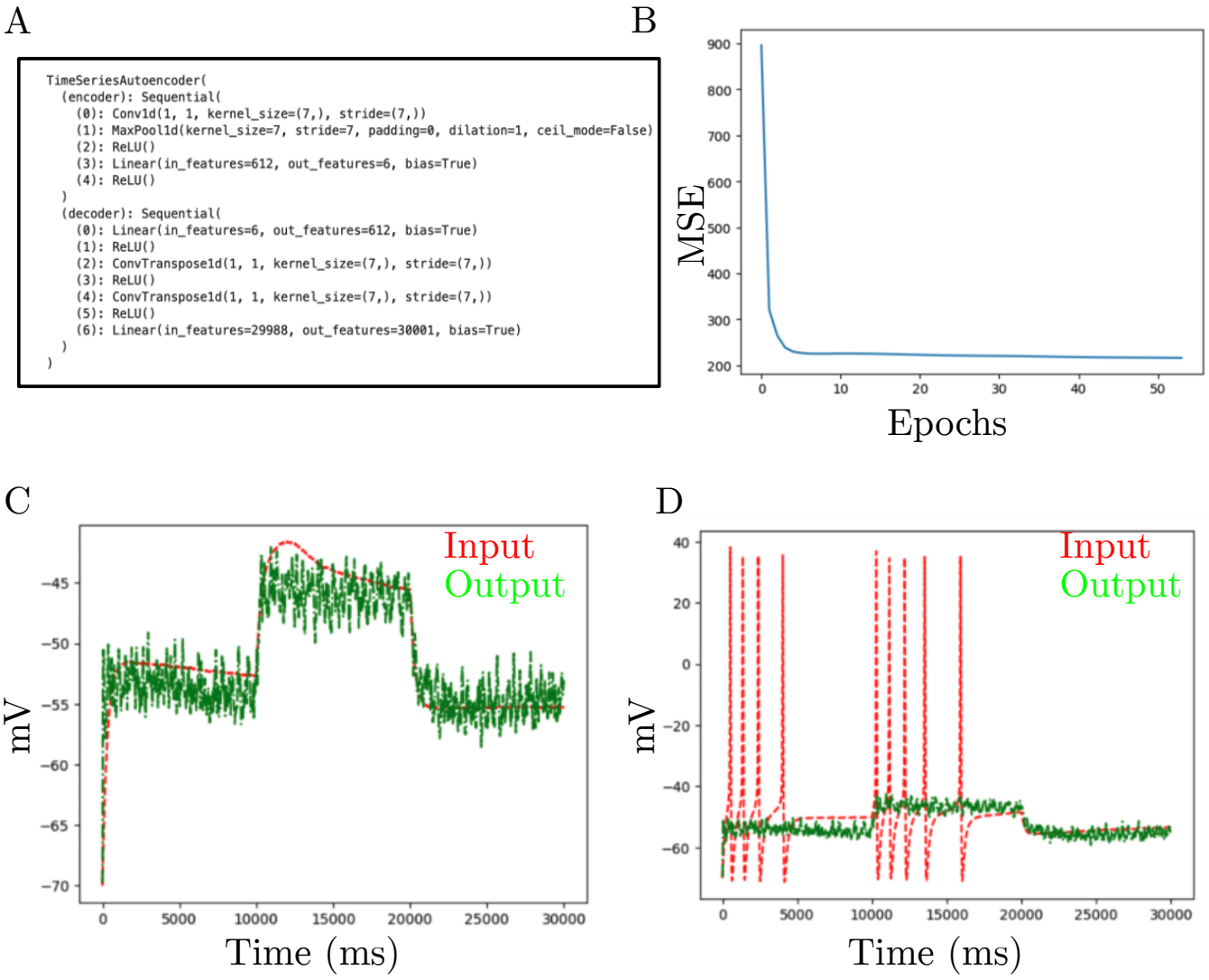
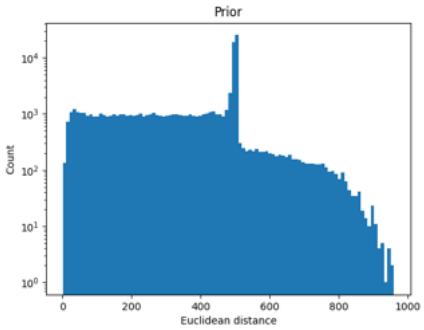


Fig. 5: **Autoencoder as a method to compute summary statistics**.
A The layered architecture we used for our autoencoder. **B** MSE between the input and the output of our autoencoder experienced an initial decrease. **C** Example trace where our autoencoder could match the input trace quite well in the sub-threshold regime. **D** Example trance where our autoencoder could not capture the spike timing or the spike shape. For C and D, the input is shown in red and the output of the autoencoder in green.

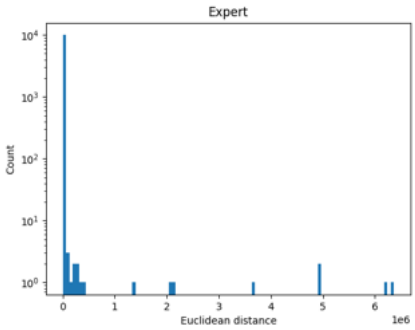
(if not considering outliers), one has to take into account that the Euclidean distances were calculated on the same expert-defined statistics used to compute SuSt. This leads to a bias that favors the expert-defined statistics approach.

Besides performance differences between the posteriors, it is also important to consider the computational resources needed to compute SuSt and to train the NDE. While training the embedding network and NDE in parallel takes more resources, this generates SuSt and a posterior at the same time. The decoupled approaches rely on first generating the SuSt and then using them to train the NDE.

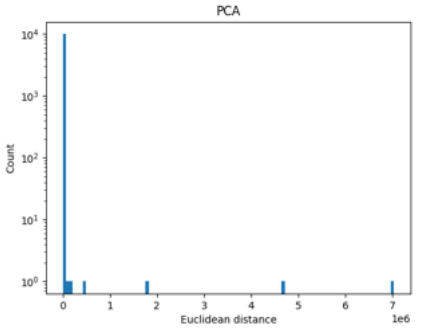
It is important to emphasize that we did not optimize the architecture (e.g., convolutional or recurrent neural networks) and hyperparameters (e.g., learning rate, learning schedule) of our embedding network. Further, we did not evaluate alternative techniques like tSNE[3], UMAP[4], or autoencoder[2]. Hence, the generalizability of our findings



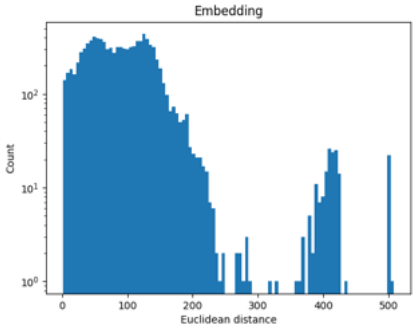
(a) Sampling from a uniform prior gives us a somewhat uniform distribution of Euclidean distances. The median is 490 with a standard deviation of 170



(b) Sampling from a posterior inferred using expert-defined statistics leads to outliers in Euclidean distance distances. The median is 9 with a standard deviation of 123,726.



(c) Sampling from a posterior inferred using PCA to compute SuSt leads to outliers in Euclidean distance distances. The median is 112 with a standard deviation of 86,660.



(d) Sampling from a posterior inferred using an embedded network to compute SuSt does not lead to outliers in Euclidean distances. The median is 89 with a standard deviation of 63.

Fig. 6: Euclidean distance reveals differences across approaches to compute SuSt. Histograms of the Euclidean distance between the set of expert-defined statistics of an example trace and those of samples drawn from the prior distributions (a, 100,000 samples) or inferred posterior distributions (b-d, 10,000 samples each).

is limited without a more comprehensive optimization and exploration of approaches to compute SuSt.

In sum, we explored the effectiveness and computational cost of computing SuSt for SBI in a case study setting. Our findings point to decoupled and coupled approaches to compute SuSt yielding posteriors that capture the underlying data while identifying tradeoffs between the presence of outliers and computational cost.

Acknowledgments

We would like to express our gratitude to the workshop organizers (Maximilian Eggl and Laura Bernaez Timon), the workshop expert speaker (Pedro J. Gonçalves), the teaching assistants (Guy Moss, Anastasia (Stasia) Krouglova), and the supporting organization (Joachim-Herz Stiftung).

References

1. Chen, Y., Gutmann, M.U., Weller, A.: Is learning summary statistics necessary for likelihood-free inference? In: Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., Scarlett, J. (eds.) Proceedings of the 40th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 202, pp. 4529–4544. PMLR (23–29 Jul 2023), https://proceedings.mlr.press/v202/chen23h.html
2. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. Science **313**(5786), 504–507 (2006). https://doi.org/10.1126/science.1127647, https://www.science.org/doi/abs/10.1126/science.1127647
3. Hinton, G.E., Roweis, S.: Stochastic neighbor embedding. Advances in neural information processing systems **15** (2002)
4. McInnes, L., Healy, J., Saul, N., Großberger, L.: Umap: Uniform manifold approximation and projection. Journal of Open Source Software **3**(29), 861 (2018). https://doi.org/10.21105/joss.00861, https://doi.org/10.21105/joss.00861