Deep Probabilistic Models for Physiological Time Series: Prediction, Generation, and Inference.

Dissertation
der Mathematisch-Naturwissenschaftlichen Fakultät
der Eberhard Karls Universität Tübingen
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

vorgelegt von
Julius Vetter
aus Esslingen

Tübingen
2025

Gedruckt mit Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Eberhard Karls Universität Tübingen.

Tag der mündlichen Qualifikation: 10.12.2025

Dekan: Prof. Dr. Thilo Stehle
1. Berichterstatter: Prof. Dr. Jakob H. Macke
2. Berichterstatter: Prof. Dr. Philipp Berens

Abstract

Physiological time series are fundamental to clinical practice and life science research. Prominent examples include heart activity recordings (electrocardiograms, or ECGs) and brain activity recordings (electroencephalograms, or EEGs). However, these time series come with several challenges. In many cases, they are noisy or contain missing values. Additionally, clinicians and researchers are often interested in inferring unobserved physiological variables from these time series, for example, when diagnosing diseases or estimating physiological parameters.

In recent years, deep probabilistic models, which use deep neural networks to model complex, high-dimensional probability distributions, have emerged as a powerful, data-driven approach to tackling many of these challenges. Despite these advances, many issues remain. For example, deep probabilistic models often require a large amount of data to accurately capture the desired distribution. For physiological time series, however, the amount of available data is often limited because datasets are subject to privacy restrictions or are expensive to obtain. Additionally, the neural networks used in deep probabilistic models often contain millions of parameters, which limits interpretability and can lead to unexpected failure cases.

To address these issues, we introduce data-efficient and interpretable deep probabilistic models for physiological time series, which can be used to predict clinical events, generate high-fidelity neurophysiological time series, and infer parameters for simulators of physiological processes.

Our first contribution concerns apnea prediction in newborn infants. We propose an interpretable model that predicts these clinical events directly from polysomnograms recorded during sleep and reveals which features of the recording are especially predictive. In our second contribution, we create data-efficient generative models of neurophysiological time series, such as EEGs, that generate realistic synthetic samples, which accurately capture many neuroscientific statistics. These models also allow for relevant time-series applications, such as imputing missing values. Finally, we propose new methods for inferring distributions of unobserved parameters from observations. Both methods concern the simulation-based setting, where a simulator (e.g. of a physiological process) is used to train a deep probabilistic model for inference. Specifically, our third contribution introduces a method for estimating high-entropy parameter distributions, which prevents the omission of valid regions in parameter space. Our fourth contribution is a method for simulation-efficient Bayesian inference that requires substantially fewer simulation data than previous approaches. We demonstrate the effectiveness of these methods on challenging parameter inference tasks involving simulators of neural voltage dynamics.

Together, the contributions in this thesis advance the applicability of deep probabilistic models for physiological time series, enabling more data-efficient and interpretable prediction, generation, and inference.

Zusammenfassung

Physiologische Zeitreihen sind für die klinische Praxis und die Lebenswissenschaften von grundlegender Bedeutung. Prominente Beispiele hierfür sind Aufzeichnungen der Herzaktivität (Elektrokardiogramme, kurz EKGs) und der Gehirnaktivität (Elektroenzephalogramme, kurz EEGs). Diese Zeitreihen sind jedoch mit einigen Herausforderungen verbunden. Oftmals sind sie verrauscht oder enthalten fehlende Werte. Darüber hinaus sind Kliniker und Forscher häufig daran interessiert, aus diesen Zeitreihen nicht beobachtete physiologische Variablen abzuleiten, beispielsweise zur Diagnose von Krankheiten oder zur Schätzung physiologischer Parameter.

In den letzten Jahren haben sich tiefe probabilistische Modelle, die tiefe neuronale Netze zur Modellierung hochdimensionaler Wahrscheinlichkeitsverteilungen nutzen, als leistungsfähiger, datengesteuerter Ansatz zur Bewältigung vieler dieser Herausforderungen bewährt. Allerdings erfordern tiefe probabilistische Modelle oft immer noch große Datenmengen, um die gewünschte Verteilung korrekt zu modellieren. Bei physiologischen Zeitreihen ist die Menge der verfügbaren Daten jedoch oft begrenzt, da Datensätze entweder Datenschutzbeschränkungen unterliegen oder ihre Beschaffung kostspielig ist. Darüber hinaus haben neuronale Netze oft Millionen von Parametern, was ihre Interpretierbarkeit einschränkt und zu unerwarteten Ausfällen führen kann.

In dieser Arbeit entwickeln wir daher dateneffiziente und interpretierbare tiefe probabilistische Modelle für physiologische Zeitreihen. Diese können zur Vorhersage klinischer Ereignisse, zur Generierung realistischer neurophysiologischer Zeitreihen sowie zur Inferenz von Parametern für Simulatoren physiologischer Prozesse verwendet werden.

Zunächst befassen wir uns mit der Vorhersage von Apnoe bei Neugeborenen. Dazu entwickeln wir ein interpretierbares Modell, das diese mithilfe von während des Schlafs aufgezeichneten Polysomnogrammen vorhersagt und dabei aufzeigt, welche Merkmale der Aufzeichnung besonders prediktiv sind. In unserem zweiten Beitrag entwickeln wir dateneffiziente generative Modelle neurophysiologischer Zeitreihen, die realistische synthetische Daten generieren, welche neurowissenschaftliche Statistiken genau widerspiegeln. Diese Modelle erlauben auch andere Anwendungen, wie beispielsweise das Schätzen fehlender Werte. Schließlich stellen wir zwei neue Methoden zur Inferenz von Verteilungen unbeobachteter Parameter vor. Beide Methoden sind Teil des simulationsbasierten Ansatzes, bei dem ein Simulator (z. B. eines physiologischen Prozesses) zum Trainieren tiefer probabilistischer Modelle verwendet wird. In unserem dritten Beitrag entwickeln wir eine Methode zur Schätzung von Verteilungen mit hoher Entropie. Dies verhindert das Auslassen wichtiger Bereiche im Parameterraum. In unserem vierten Beitrag präsentieren wir eine simulations-effiziente Bayes'sche Inferenzmethode, für die wesentlich weniger Simulationsdaten erforderlich sind als bei bisherigen Ansätzen. Die Wirksamkeit beider Methoden demonstrieren wir anhand der Inferenz von Parametern von Simulatoren aus der Neurobiologie.

Die Beiträge dieser Arbeit tragen gemeinsam dazu bei, die Anwendbarkeit tiefer probabilistischer Modelle für physiologische Zeitreihen weiter voranzubringen, indem sie dateneffizientere und besser interpretierbare Vorhersage, Generierung und Inferenz ermöglichen.

Acknowledgments

I have to thank many people, without whom this PhD would not have been possible.

First and foremost, thank you, Jakob, for your supervision and guidance over the past four years. No matter what direction my research took, I could always count on your clarity about what truly matters, and I learned a great deal from that.

Thank you, Richard, for being the best postdoc supervisor imaginable. I will never forget our hour-long walks and discussions.

Thank you also to the rest of my PhD committee, my second examiner Philipp, as well as Katharina and Oliver.

Thank you to everyone in the Mackelab. Thank you, Guy, Auguste, Sebastian, Zina, Manuel, Jai, Matthijs, Linda, Michael, Poornima, Jan, Daniel, Janne, Cornelius, Pedro, Lucas, Lisa, Isaac, Stefan, Nicolas. I'm thankful for every lunch we shared, for every scientific and non-scientific discussion, for the great collaborations, for the support and encouragement, and for so much more. You all made this such a great place to do research. Thank you, Franzi, for your making these years so easy from an administrative point of view.

Finally, thank you to my friends, my parents Simone and Stefan, and my siblings Franka and Henri for their unwavering support. And last, thank you, Toni, for everything.

List of Publications

Primary Publications

1. **Julius Vetter**, Kathleen Lim, Tjeerd M.H. Dijkstra, Peter A. Dargaville, Oliver Kohlbacher, Jakob H. Macke, and Christian F. Poets. *Neonatal apnea and hypopnea prediction in infants with Robin sequence with neural additive models for time series.* PLOS Digital Health 3, no. 12 (2024).

2. **Julius Vetter**, Jakob H. Macke*, and Richard Gao*. *Generating realistic neurophysiological time series with denoising diffusion probabilistic models.* Patterns 5, no. 9 (2024).

3. **Julius Vetter***, Guy Moss*, Cornelius Schröder, Richard Gao, and Jakob H. Macke. *Sourcerer: Sample-based maximum entropy source distribution estimation.* Advances in Neural Information Processing Systems 37, (2024).

4. **Julius Vetter***, Manuel Gloeckler*, Daniel Gedon, and Jakob H. Macke. *Effortless, simulation-efficient Bayesian inference using tabular foundation models.* Advances in Neural Information Processing Systems 38, (2025).

Supporting Publications

1. Jaivardhan Kapoor*, Auguste Schulz*, **Julius Vetter**, Felix Pei, Richard Gao, and Jakob H. Macke. *Latent diffusion for neural spiking data.* Advances in Neural Information Processing Systems 37, (2024).

2. Auguste Schulz, **Julius Vetter**, Richard Gao, Daniel Morales, Victor Lobato-Rios, Pavan Ramdya, Pedro J. Gonçalves, and Jakob H. Macke. *Modeling conditional distributions of neural and behavioral data with masked variational autoencoders.* Cell Reports 44, no. 3 (2025).

* denotes equal contribution.

Contents

Chapter 1

Introduction

Physiological measurement, that is, the measurement of biological functions or processes within a living organism, is fundamental to research in the life sciences, as well as clinical practice. Physiological measurements allow researchers to develop or confirm hypotheses about the underlying physiological systems and enable clinicians to monitor patients' health and diagnose disease. They often come in the form of time series, where different measurements are collected at subsequent points in time. In clinical practice, important examples of such physiological time series include the electrical activity of the heart, as measured by an electrocardiogram (ECG), or the spontaneous electrical activity of the brain, as measured by an electroencephalogram (EEG) [1, 2]. In the life sciences, physiological time series are even more varied and differ substantially between subfields. Taking neuroscience as an example, neuroscientists measure neurophysiological time series at various resolutions. Examples include recordings of action potentials from individual neurons as well as local field potentials (LFPs), which capture the aggregate activity of neuronal populations [3].

After recording, the main challenge for researchers and clinicians is to extract scientific or clinical insights from these physiological time series. This usually involves estimating unobserved variables associated with the underlying physiological system. In some cases, these unobserved variables are discrete. Examples include the detection and prediction of disease or clinical events. In other cases, unobserved variables follow a more complex, high-dimensional and continuous distribution: In the life sciences, researchers routinely create or use simulators of physiological systems. These simulators often produce a physiological time series based on a set of parameters values. However, it is typically difficult to determine which parameter values produce a time series with the desired features. Therefore, researchers are interested in inferring distributions of feasible values from observed data—a problem known as parameter inference. The difficulty of estimating distributions of unobserved variables from physiological time series is exacerbated by the fact that they are often noisy and contain missing values [4]. This is particularly true in clinical settings, which lack the controlled environment of a scientific experiment. Therefore, these time series often need to be processed before they can be used. Common processing tasks include imputing missing values, detecting outliers and removing noise. For all these reasons, extracting scientific or clinical insights from physiological time series is challenging and often requires in-depth domain expertise and specialized approaches. The growing availability of increasingly high-dimensional physiological data in recent years has made this challenge even more difficult, necessitating new, more efficient approaches [5, 6].

Today, machine learning provides a powerful approach to capture intricate relationships directly from data. Specifically, probabilistic machine learning [7] allows us to train deep probabilistic models that can approximate complex (conditional) probability distributions using expressive parame-

terizations based on deep neural networks [8]. Along with the development of neural architectures tailored to time series—spanning from classical recurrent neural networks [9, 10] and temporal convolutional neural networks [11, 12] to modern architectures, such as structured state space models [13, 14]—deep probabilistic models are now used for many physiological time series applications. Deep time-series classifiers [15] can discriminate between different physiological states, or detect and predict disease and clinical events, such as seizures or heart arrhythmias [16–18]. Furthermore, deep generative models can generate synthetic physiological time series or probabilistically impute missing values within them [19–23]. In addition, deep generative models are used to estimate complex and high-dimensional distributions of unobserved variables associated with physiological time series, may it be for parameter inference, or the extraction of latent dynamics [24–28].

Although deep probabilistic models have been successfully applied to physiological time series, many issues remain. For example, when time series are high-dimensional or depend on unobserved variables in a complex manner, these models may require a substantial amount of training data to accurately capture the desired distribution [29]. This issue is particularly pressing in cases where high-quality data is scarce and expensive to obtain, as is often the case in clinical settings. Addressing this issue requires developing data-efficient deep probabilistic models that can attain accurate estimates with limited training data. Another issue with using these models to estimate unobserved variables is that they obscure the time series features relevant to their prediction, for instance, when predicting clinical events [30, 31]. However, in a high-stakes clinical environment, it is crucial that models provide insight into how they reach a given conclusion. Such model interpretability allows clinicians to trust or distrust the model and make informed decisions about patients' treatment [32]. Furthermore, when estimating high-dimensional continuous distributions of unobserved variables, the space of possible solutions can be very complicated. In such cases, deep probabilistic models often collapse to degenerate estimates due to their flexible parameterization [33]. This is problematic when the estimated distributions are used to obtain clinical or scientific insight because important structure may be missed. To address this issue, it is necessary to constrain or regularize the model estimates to ensure their interpretability.

In this thesis, we address these issues by developing *deep probabilistic models for physiological time series* that enable the interpretable *prediction* of clinical events, the data-efficient *generation* of densely sampled and multivariate time series, as well as improved data efficiency and non-degenerate estimates in parameter *inference*.

In our first publication, we develop a deep probabilistic model to predict apneas in newborn infants. Neonatal apneas, which are temporary cessations of breathing during sleep, can seriously affect an infant's healthy development. Automatically predicting these events would make monitoring these infants less labor-intensive and improve the quality of care [34]. However, previous work on this application is sparse and has mostly focused on apnea detection [35–37]. Therefore, we develop an interpretable deep probabilistic model that uses polysomnography recordings—a collection of physiological time series recorded during sleep including modalities related to breathing, blood oxygen, and heart activity—to predict upcoming neonatal apneas. Our model makes accurate predictions based directly on these physiological time series and requires only minimal preprocessing. Importantly, our model is inherently interpretable, enabling clinicians to determine which modalities and features of the polysomnography recording were relevant to the model's prediction.

Beyond prediction, it is often desirable to directly model the distribution of recorded physiological time series by training a deep generative model. This is because, apart from the generation of synthetic time series, a generative model can also be used to address many other challenges, such as

imputation or classification. In recent years, diffusion models have become the predominant type of deep generative model [38]. While previous work has demonstrated the effectiveness of diffusion models for shorter time series, their neural architectures are insufficient for densely sampled and multivariate nature of neurophysiological time series [19, 39, 40]. Thus, in our second publication, we develop data-efficient diffusion models that can accurately model neurophysiological time series, such as EEG and LFP. To achieve this, we incorporate neuroscientific domain knowledge into the diffusion process and build a tailored neural architecture to capture complex and long-range temporal dependencies. We demonstrate that the synthetic neurophysiological times series generated from our diffusion models accurately capture features and statistics that are important to neuroscientists, such as sharp wave ripples and phase-amplitude couplings. Our diffusion models can also be applied to various neuroscientific applications such as imputing missing time series channels, detecting outliers, and classifying brain states.

Finally, we develop deep probabilistic models to estimate high-dimensional and continuous distributions of unobserved variables associated with physiological time series. More precisely, this thesis considers the aforementioned problem of parameter inference: When studying physiological systems and their associated time series, researchers often create parameterized models, or simulators, of these systems. The simulator's underlying parameters are typically of great scientific or clinical interest, yet difficult or even impossible to measure. Therefore, the goal of parameter inference is, in broad terms, to solve the following problem: Given one or multiple observed physiological time series, what is the distribution of feasible simulator parameters that, when used for simulation, will reproduce the observed time series? In this thesis, we consider two approaches to perform parameter inference with slightly different objectives: Bayesian inference and source distribution estimation.

Bayesian inference uses Bayes' theorem to estimate the posterior distribution of (simulator) parameters given one or more observations. One modern approach to Bayesian inference is simulation-based inference (SBI) [41]. SBI is used to estimate the posterior distribution in challenging settings involving complex simulators where traditional Bayesian inference methods cannot be used. Specifically, neural SBI methods approximate the posterior distribution using a deep probabilistic model trained on a dataset of parameter-simulation pairs [42–44]. A key challenge for SBI is simulation efficiency. Since simulators can be computationally expensive, it is crucial to accurately infer the posterior with as few simulations as possible. To address this challenge, we repurpose a probabilistic foundation model for regression and classification to enable training-free and simulation-efficient SBI. Using a set of established benchmark tasks, we demonstrate that our approach is highly simulation-efficient. In addition, we apply it to challenging parameter inference problem in neuroscience: We infer parameters of the Hodgkin-Huxley model, which describes how action potentials in neurons are initiated and propagated, given real recordings of action potentials. Our approach requires orders of magnitude fewer simulations to obtain high-quality posteriors than baseline SBI methods.

In contrast to Bayesian inference, source distribution estimation aims to estimate a shared distribution of different simulator parameters that generate simulations closely matching a whole dataset of observations. In general, due to the lack of prior constraints, source distributions are not unique, making the problem ill-posed. While, similar to neural SBI methods, recent work has developed approaches which use deep probabilistic models to estimate increasingly complex source distributions [45–47], the ill-posedness of source distribution estimation has largely been ignored. In our fourth publication, we propose a method that addresses this ill-posedness by estimating source distributions that have high entropy and thus retain as much uncertainty as possible. On a set of benchmark tasks, we demonstrate that our method is able to estimate source distributions

with substantially higher entropy than previous approaches. We again use our method for parameter inference in the Hodgkin-Huxley model and estimate a high-entropy source distribution of feasible parameters given a dataset of action potential recordings.

The thesis is organized as follows. Chapter 2 provides general background on deep probabilistic models for physiological time series. Specifically, in Sec. 2.1, I give an overview over deep probabilistic models and their applications in parameter inference, introducing both (simulation-based) Bayesian inference and source distribution estimation. In Sec. 2.2, I introduce neural architectures that are tailored to handle times series. Chapter 3 then gives an overview of the four main publications of this thesis. Finally, in Chapter 4, I conclude with a discussion of our results and directions for future work. In summary, this thesis makes further advances in the application of deep probabilistic models to physiological time series, enabling interpretable prediction, data-efficient and high-fidelity generation as well as more simulation-efficient and well-posed parameter inference.

Chapter 2

Background

This chapter provides general background relevant to the work presented in this thesis. First, I introduce deep probabilistic models and their applications in parameter inference (Sec. 2.1). Then, I provide an overview of the most common deep neural network architectures used in deep probabilistic models for time series (Sec. 2.2).

2.1 Deep probabilistic models

Deep probabilistic models aim to approximate a typically intractable or unknown probability distribution $p(\boldsymbol{x})$ using a flexibly parameterized, tractable probability distribution $q_\phi(\boldsymbol{x})$, where ϕ are the model parameters [7]. Flexibly parameterized here usually means parameterized by a deep neural network, and I discuss possible options for time series in Sec. 2.2. More often than not, the probability distribution of interest is conditional: Given a joint distribution $p(\boldsymbol{x}, \boldsymbol{y})$ over two covariates $\boldsymbol{x}$ and $\boldsymbol{y}$, the goal is to model $p(\boldsymbol{y} \mid \boldsymbol{x})$, that is the distribution of $\boldsymbol{y}$ given $\boldsymbol{x}$ is known or observed.

After implementing a deep probabilistic model, its parameters ϕ need be to chosen such that $q_\phi(\boldsymbol{x}) \approx p(\boldsymbol{x})$ or, in the conditional case, $q_\phi(\boldsymbol{y} \mid \boldsymbol{x}) \approx p(\boldsymbol{y} \mid \boldsymbol{x})$ (both cases are largely equivalent from a technical perspective, and I switch between them depending on the context). Obtaining such feasible model parameters is typically achieved by defining a distance metric or divergence D between the model q_ϕ and the target distribution p, and choosing parameters ϕ^* that minimize D:

$$\phi^* = \arg\min_\phi D(p, q_\phi). \tag{2.1}$$

In general, it can be challenging to make the optimization problem in Eq. (2.1) tractable. This is because we usually do not know p and only have access to it in the form of a (typically finite) dataset of samples $\mathcal{D} = \{\boldsymbol{x}_i\}_{i=1}^n$, which are assumed to be independently drawn from $p(\boldsymbol{x})$. Additionally, the distance metric or divergence D may be difficult compute, or the probability density of our deep probabilistic model q_ϕ may itself not be directly accessible. Models in which the density can be directly evaluated are called explicitly parameterized. In contrast, models with an inaccessible density are called implicitly parameterized.

When a deep probabilistic model is explicitly parameterized, the standard approach to fitting it is maximizing the (log-)likelihood $\log q_\phi(\boldsymbol{x})$ under the target distribution p:

$$\phi^* = \arg\max_\phi \mathbb{E}_{\boldsymbol{x}\sim p}[\log q_\phi(\boldsymbol{x})] \approx \arg\max_\phi \mathbb{E}_{x\sim\mathcal{D}}[\log q_\phi(\boldsymbol{x})]. \tag{2.2}$$

Here, $\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[\cdot]$ indicates that we approximate the expectation under p with the empirical data distribution given by the (finite) dataset $\mathcal{D}$.

Notably, this maximization is equivalent to selecting the (forward) KL-divergence as the discrepancy metric D in Eq. (2.1) and minimizing it:

$$\arg\max_{\phi} \mathbb{E}_{\boldsymbol{x}\sim p}[\log q_\phi(\boldsymbol{x})] = \arg\min_{\phi} D_{\mathrm{KL}}(p\|q_\phi), \quad \text{where } D_{\mathrm{KL}}(p\|q) = \mathbb{E}_{x\sim p}\left[\log \frac{p(\boldsymbol{x})}{q(\boldsymbol{x})}\right]. \tag{2.3}$$

This equivalence holds because the KL-divergence can be written as the sum of the cross entropy $H(p,q)$ between distributions p and q and the negative entropy $H(p)$ of p, that is,

$$D_{\mathrm{KL}}(p\|q) = H(p,q) - H(p) = \mathbb{E}_{\boldsymbol{x}\sim p}\left[-\log q(\boldsymbol{x})\right] - \mathbb{E}_{\boldsymbol{x}\sim p}\left[-\log p(\boldsymbol{x})\right]. \tag{2.4}$$

Since the entropy term in Eq. (2.4) only depends on q, it can be ignored during optimization. As a result, only the cross-entropy term remains to be minimized, which is the negation of Eq. (2.2).

Sec. 2.1.1 and Sec. 2.1.2 introduce two types of explicitly parameterized deep probabilistic models trained with maximum likelihood: deep classifiers and normalizing flows. For implicitly parameterized models, approximations to the original objective in Eq. (2.1) are necessary. One typical such approximation is the evidence lower bound (ELBO), which as its name suggests, is a tractable lower bound to the (log-)likelihood of the data under the model (evidence). I will introduce the ELBO in the context of diffusion models in Sec. 2.1.2. There are many other ways to set up or optimize implicitly parameterized models, and a comprehensive discussion is beyond the scope of this thesis. Just one example is the use of a purely sample-based distance metrics or divergences D, such as maximum mean discrepancy (MMD, [48]), sliced-Wasserstein distance [49] or implicitly defined adversarial divergences [50].

2.1.1 Deep classifiers

The goal of classification is to model a discrete distribution over a set of categories or labels given an object to be classified. Formally, we want to learn a conditional model $q_\phi(y \mid \boldsymbol{x})$ that approximates the discrete distribution $p(y \mid \boldsymbol{x})$ over a finite number of elements $y \in \{1, \dots, C\}$ given some condition $\boldsymbol{x}$. Often, there are only two categories or labels ($C = 2$), which is known as binary classification.

Because discrete distributions are defined over a finite set of elements, ensuring that the model represents a properly normalized probability distribution is straightforward: The neural network outputs $f_\phi(\boldsymbol{x})_j$ for label j—also-called logits—are simply re-normalized using the softmax function

$$q_\phi(y \mid \boldsymbol{x}) = \frac{\exp(f_\phi(\boldsymbol{x})_y)}{\sum_{j=1}^{C} \exp(f_\phi(\boldsymbol{x})_j)}, \tag{2.5}$$

which ensures that the probabilities associated with each logit are nonnegative and sum to one.

Constructing the discrete distribution in this way yields a tractable likelihood. Thus, deep classifiers are an instance of explicitly parameterized deep probabilistic models and can be optimized by maximizing that likelihood. For classifiers, this is usually referred to as minimizing the cross-entropy loss

$$\mathcal{L}_{CE}(\phi) = \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}}\left[-\log q_\phi(y \mid \boldsymbol{x})\right] = \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}}\left[-\log\left(\frac{e^{f_\phi(\boldsymbol{x})_y}}{\sum_{j=1}^{C} e^{f_\phi(\boldsymbol{x})_j}}\right)\right]. \tag{2.6}$$

Deep classifiers were the first successful deep probabilistic models. In 2012, AlexNet [51], a deep classifier for images, won the ImageNet challenge [52], ushering in the era of modern deep learning. Today, deep classifiers are used extensively across virtually all machine learning application domains, including those related to physiological time series [16–18, 53, 53, 54].

2.1.2 Deep generative models

Unlike modeling univariate and discrete distribution, such as in classification, approximating high dimensional and continuous probability distributions (conditional or unconditional) is considerably more challenging, and has been the main focus of the field of deep generative modeling. Providing a complete overview of deep generative modeling is beyond the scope of this thesis. Therefore, I will only discuss the generative models relevant to this thesis: normalizing flows and diffusion models. Other common types of deep generative models, such as variational autoencoders [55] and generative adversarial networks [50], will be skipped.

Normalizing flows

Normalizing flows [56, 57] are a widely used class of explicitly parameterized deep generative models. The idea of normalizing flows is to transform a base distribution $\pi(\boldsymbol{z})$, typically an isotropic Gaussian $\mathcal{N}(0, I)$, into (an approximation of) the target distribution $p(\boldsymbol{x})$ using a sequence of invertible transformations $\boldsymbol{x} = f(\boldsymbol{z})$. Specifically, the inverse $\boldsymbol{z} = f^{-1}(\boldsymbol{x})$ of these transforms is designed to be both tractable and differentiable. This invertibility allows one to explicitly evaluate the density after transformation using the change of variables formula:

$$p(\boldsymbol{x}) = \pi(\boldsymbol{z}) \det \left| \frac{\partial \boldsymbol{z}}{\partial \boldsymbol{x}} \right| = \pi(f^{-1}(\boldsymbol{x})) \det \left| \frac{\partial f^{-1}(\boldsymbol{x})}{\partial \boldsymbol{x}} \right|. \tag{2.7}$$

Here, the determinant of the Jacobian matrix $\frac{\partial f^{-1}(\boldsymbol{x})}{\partial \boldsymbol{x}}$ accounts for how volumes are stretched or compressed by the transformation. Normalizing flows are then simply a composition of several (parameterized) invertible transformations in sequence:

$$q_\phi(\boldsymbol{x}) = f_{\phi_N}(f_{\phi_{N-1}}(...(f_{\phi_1}(\boldsymbol{z})))). \tag{2.8}$$

Through repeated application of the change-of-variables formula, the overall density $q_\phi(\boldsymbol{x})$ remains tractable, and parameters can be fitted by maximizing the likelihood of the observed data (Eq. (2.2)). There are many ways to construct these invertible transformations, and this remains an active area of research [58–61]. While this invertibility constraint makes normalizing flows tractable and easy to use, it also restricts their expressiveness and limits the complexity of the distributions they can model. Normalizing flows particularity struggle when the distributions are high-dimensional (e.g., long and multivariate physiological time series).

Although so far I only discussed unconditional normalizing flows, modeling a conditional distribution $p(\boldsymbol{x} \mid \boldsymbol{y})$ is straightforward. It only requires changes to the construction of f to allow for a conditional input $\boldsymbol{y}$. As in the unconditional case, the conditional normalizing flow is then trained by maximizing the conditional log likelihood, $\mathbb{E}_{\boldsymbol{x},\boldsymbol{y}\sim p(\boldsymbol{x},\boldsymbol{y})}[\log q_\phi(\boldsymbol{x} \mid \boldsymbol{y})]$.

Generative models, such as conditional normalizing flows, have many applications beyond generating synthetic samples: For example, suppose we have trained a conditional normalizing flow to generate a physiological time series based on a discrete disease state $y \in \{0, 1\}$. Due to the

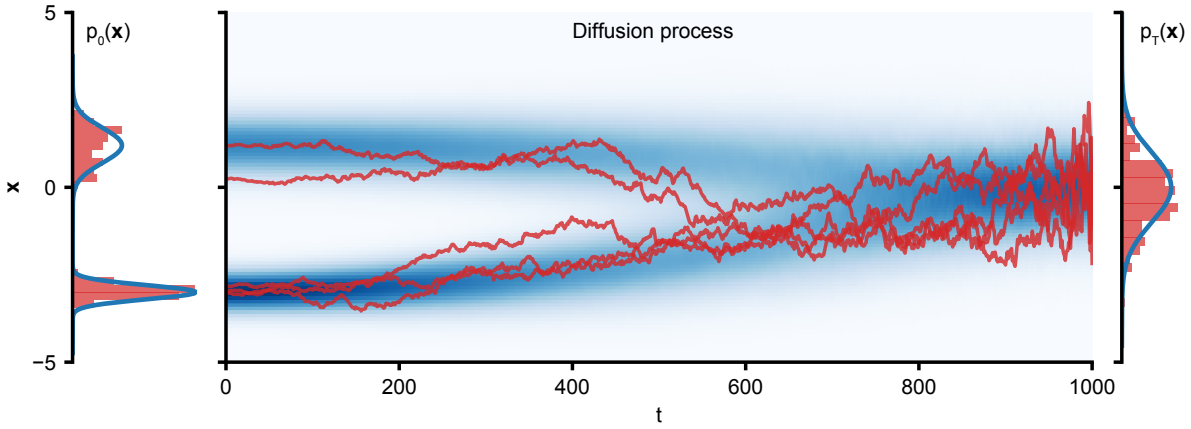


Figure 2.1: Forward diffusion process of Denoising Diffusion Probabilistic Models (DDPMs). The bimodal target distribution $p_0(\boldsymbol{x})$ gets transformed into the standard Gaussian distribution by gradually adding Gaussian noise over T diffusion timesteps t. For this visualization, we set $T = 1000$, which is a common choice in practice. The target and base densities at $t = 0$ and $t = T$ are shown in blue. The marginal densities for the steps in between are visualized by the blue heatmap. Samples from the target and base distributions, as well as five example diffusion paths are shown in red. The central idea behind DDPMs is to learn an approximation of the reverse process from $t = T$ to $t = 0$.

tractable densities, the same flow can be used for classification via Bayes's theorem:

$$p(y = 1 \mid \boldsymbol{x}) \approx \frac{q_\phi(\boldsymbol{x} \mid y = 1)}{0.5q_\phi(\boldsymbol{x} \mid y = 1) + 0.5q_\phi(\boldsymbol{x} \mid y = 0)}, \tag{2.9}$$

here assuming the prior $p(y)$ places equal probability on both disease states [62].

Diffusion models

In recent years, diffusion models [38, 63] have become the predominant type of deep generative model. Diffusion models are generally more expressive than normalizing flows because they permit the use of unconstrained, non-invertible neural networks. Nevertheless, diffusion models are similar to normalizing flows in that they gradually transform a known base distribution π into the target distribution p. However, while normalizing flows achieve this using a composition of a few invertible transformations, diffusion models aim to model a fine grained, even continuous transformation between the base and target distributions. To accomplish this, direct access to a tractable density that could be used for maximum likelihood training must be given up. Thus, diffusion models are an example of an implicitly parameterized deep generative model.

Below, I introduce Denoising Diffusion Probabilistic Models (DDPMs) [64], a common formulation of diffusion models. The main idea behind DDPMs is to gradually corrupt data sampled from the target distribution by adding Gaussian noise and then learn a reverse process that undoes this corruption. Since the fully corrupted data is indistinguishable from Gaussian noise, access to the reverse process enables sampling from the target distribution by initiating the process with Gaussian noise followed by gradual denoising.

In a DDPM, we consider the forward process that is specified by the following Markov chain:

$$p(\boldsymbol{x}_{1:T} \mid \boldsymbol{x}_0) = \prod_{t=1}^{T} p(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}), \quad \text{where} \quad p(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{1-\beta_t}\boldsymbol{x}_{t-1}, \beta_t I). \tag{2.10}$$

Starting from clean data $\boldsymbol{x}_0 \sim p(\boldsymbol{x}_0) = p_0(\boldsymbol{x})$, the forward process gradually transforms the data into white noise (Fig. 2.1). Due to the Gaussianity and careful construction of the noise schedule $\beta_1, \ldots, \beta_T$, the marginal distributions $p(\boldsymbol{x}_t \mid \boldsymbol{x}_0)$ admit a closed-form representation

$$p(\boldsymbol{x}_t \mid \boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0, (1-\bar{\alpha}_t)I), \quad \text{with} \quad \bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s). \tag{2.11}$$

The goal is now to learn the reverse process

$$q_\phi(\boldsymbol{x}_{0:T}) = p_T(\boldsymbol{x}) \prod_{t=1}^{T} q_\phi(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t), \quad \text{with} \quad q_\phi(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{x}_{t-1}; \mu_\phi(\boldsymbol{x}_t, t), \Sigma_\phi(\boldsymbol{x}_t, t)). \tag{2.12}$$

Here, the mean μ_ϕ is parameterized by a deep neural network and the variance is set to $\Sigma_\phi(\boldsymbol{x}_t, t) = \sigma_t^2 I$ with $\sigma_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$. Importantly, this formulation does not allow for the direct maximization of the log-marginal-likelihood $\log q_\phi(\boldsymbol{x}_0)$. Instead, DDPMs are optimized using a variational lower bound on $\log q_\phi(\boldsymbol{x}_0)$. This so-called evidence lower bound (ELBO) provides a tractable expression in terms of the latent variables of the diffusion process, that is, the noised data at different levels of corruption $\boldsymbol{x}_{1:T}$:

$$-\log q_\phi(\boldsymbol{x}_0) \leq \mathbb{E}_{\boldsymbol{x}_{1:T} \sim p(\boldsymbol{x}_{1:T} \mid \boldsymbol{x}_0)} \left[-\log \frac{q_\phi(\boldsymbol{x}_{0:T})}{p(\boldsymbol{x}_{1:T} \mid \boldsymbol{x}_0)} \right]. \tag{2.13}$$

In practice, the above is simplified by reparameterizing the forward process. Rather than predicting the mean of $\boldsymbol{x}_{t-1}$, the network is trained to predict the noise $\boldsymbol{\epsilon}$ that was added at each timestep. After dropping a few weighting terms, one arrives at the simplified objective

$$\mathcal{L}_{\text{simple}}(\phi) = \mathbb{E}_{x_0 \sim \mathcal{D}, \boldsymbol{\epsilon}, t} \left[\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\phi(x_t, t)\|^2 \right], \quad \text{with} \quad x_t = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, \tag{2.14}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(0, I)$ and $t \sim \text{DiscreteUniform}(1, T)$. This simplified objective can be optimized across all timesteps using a single neural network, typically referred to as the denoiser or diffusion backbone. Timesteps t are simply passed in as an additional input.

After training, samples from the diffusion model can be generated by iteratively denoising samples starting from Gaussian noise. However, sampling from a diffusion model is slower than sampling from a normalizing flow because it requires many sequential evaluations of the denoiser. Consequently, significant efforts have been devoted to accelerating the sampling process by enhancing diffusion solvers, or distilling them into faster variants such as consistency models [65–68]. It is also worth noting that DDPMs are just a special instance of a broader class of score-based diffusion models, which are typically expressed as a stochastic differential equation involving the score $\nabla \log p(\boldsymbol{x})$ [69]. Finally, continuous normalizing flows, which are a continuous generalization of previously introduced (discrete) normalizing flows, present a closely related, and in some cases equivalent, class of models [70].

Today, diffusion models are the predominant type of deep generative model. They are best known for their ability to generate high-quality images and videos, a capability that has received significant attention from the general public [71, 72]. Beyond these well-known applications, diffusion models are used in many areas of machine learning, including time series-related applications, such as forecasting and imputation [19, 20, 40].

2.1.3 In-context learning using prior-data fitted networks

To approximate a given (conditional) distribution $p(\boldsymbol{y} \mid \boldsymbol{x})$, for which only joint samples $\mathcal{D} = \{\boldsymbol{y}_i, \boldsymbol{x}_i\}_{i=1}^n$ are available, deep probabilistic models are typically optimized or trained with respect to objectives (e.g. Eq. (2.2), Eq. (2.6), or Eq. (2.14)) computed on those samples. This training process needs to be repeated for a different target distribution $p'(\boldsymbol{y} \mid \boldsymbol{x})$ with different dataset of samples $\mathcal{D}' = \{\boldsymbol{y}_i', \boldsymbol{x}_i'\}_{i=1}^n$.

With the rise of large language models (LLMs, 73), a new paradigm—in-context learning (ICL)—has emerged. In ICL, the training process is replaced by a single forward pass through a larger, pretrained model. That is, rather than performing optimization for a specific task or training dataset, the task description or dataset is provided to the model as context. The model then adapts its output distribution based on the information present in this context. As mentioned, the most prominent example of in-context learners are LLMs, which can solve many different tasks depending on the language prompt they are conditioned on. Here, I discuss another type of in-context learner: prior-data fitted networks (PFNs) [74]. PFNs primarily focus on modeling numerical and continuous distributions rather than the discrete sequences of language tokens modeled by LLMs.

The key to making ICL work is pretraining. For PFNs, a (transformer-based, 75) deep probabilistic model is pretrained on a large set of synthetic datasets sampled from a structured prior distribution over datasets. If the prior distribution is constructed properly and effectively captures the space of "interesting" or "relevant" distributions, the trained PFN is able to approximate the target distribution trough inferring the desired distribution from the context dataset.

To make the above more precise, we consider the case of training a PFN to perform in-context classification [76]. Specifically, we aim to model $p(y \mid \boldsymbol{x})$ where $y \in \{1, \dots, C\}$ by conditioning the PFN on a dataset of joint samples $\{y_i, \boldsymbol{x}_i\}_{i=1}^n$ from $p(y, \boldsymbol{x})$. After constructing the prior $\Pi(p)$ over joint probability distributions $p(y, \boldsymbol{x})$, creation of the pretraining dataset proceeds in two steps. In the first step, N distributions p_j are sampled from the prior. In the second step, $n+1$ samples $y_i, \boldsymbol{x}_i$ are drawn from each distribution p_j. This two-step process results in a "dataset of datasets"

$$\mathcal{G} = \left\{ (y_0^{(j)}, \boldsymbol{x}_0^{(j)}) \cup D^{(j)} \right\}_{j=1}^N, \quad \text{where } D^{(j)} = \{y_i^{(j)}, \boldsymbol{x}_i^{(j)}\}_{i=1}^n. \tag{2.15}$$

The PFN is then trained on $\mathcal{G}$ by maximizing the model's log-likelihood, that is, by minimizing the usual cross entropy loss (Eq. (2.6)), which here takes the form

$$\mathcal{L}(\phi)_{PFN} = \mathbb{E}_{y_0, \boldsymbol{x}_0, \mathcal{D} \sim \mathcal{G}} \left[-\log q_\phi \left(y_0 \mid \boldsymbol{x}_0, \mathcal{D} \right) \right]. \tag{2.16}$$

After pretraining, an unseen context dataset D_o together with test point x_o can be passed to the PFN to predict y. Formally, this pretraining process corresponds to approximating the target distribution $p(y \mid \boldsymbol{x})$ with the Bayesian posterior predictive distribution:

$$p(y \mid \boldsymbol{x}, \mathcal{D}) = \int p(y \mid \boldsymbol{x}) \, d\Pi(p \mid \mathcal{D}), \tag{2.17}$$

where $\Pi(p \mid \mathcal{D})$ indicates the posterior distribution over distributions p given a dataset $\mathcal{D}$. Hence, the basic idea of PFNs—training a deep probabilistic model using many synthetic datasets from a "dataset simulator" with the goal of amortizing inference over datasets—can be seen as an instance of amortized (simulation-based) Bayesian inference (see Sec. 2.1.4 for more details).

In practice, PFNs are trained using billions of synthetic datasets with various different sizes [77, 78]. The prior over these datasets is usually realized via random structural causal models [79].

Regression models are also straightforward to implement by using an explicitly parameterized density, such as a (one-dimensional) normalizing flow (Sec. 2.1.2). For lower-dimensional numerical problems and tabular data, PFNs have proven to be very powerful. The state-of-the-art PFN models for tabular classification and regression outperform many classical methods, such as boosting, while being completely training-free [78, 80]. PFNs are especially powerful in the low-data regime, where only a few samples are available for training. While classical methods may easily overfit, PFNs can handle small training datasets gracefully by sidestepping the brittle optimization process. Furthermore, by modeling the prior predictive distribution, PFNs implicitly account for uncertainty in the target density by providing a weighted average of distributions that are likely given the context dataset.

2.1.4 Applications of deep probabilistic models for parameter inference

In this section, I introduce applications of deep probabilistic models for machine learning-based parameter inference. Given a model or simulator $s : \boldsymbol{\theta} \mapsto \boldsymbol{x}$ (e.g. of a physiological process) mapping from parameter space to observation space, the goal of parameter inference is to solve the inverse direction $\boldsymbol{x} \mapsto \boldsymbol{\theta}$. Inferring simulator parameters from observations is a fundamental problem in science, relevant well beyond its application to simulators for physiological processes. This is because these parameters are often of major scientific interest but very difficult or even impossible to measure. In such cases, the only way to learn about them is to infer their values from observed data together with a model or simulator. Other example applications for parameter inference problems range from black-hole physics to genetics [81–84].

The mapping between the parameter and observation spaces, as described by the simulator s, is usually not deterministic. This is because the simulators of, in our case, physiological processes may have inherent random components, or the available observations may be corrupted by measurement noise. If, for a given parameter $\boldsymbol{\theta}$, different outcomes are possible, the simulator can be seen as a conditional probability distribution $p(\boldsymbol{x} \mid \boldsymbol{\theta})$ known as the likelihood. Because $p(\boldsymbol{x} \mid \boldsymbol{\theta})$ is non-deterministic, it is typically impossible to obtain a deterministic mapping for the inverse direction. Instead, the mapping $\boldsymbol{x} \mapsto \boldsymbol{\theta}$ must also be described by a probability distribution. In the following, I discuss two approaches to parameter inference, Bayesian inference and source distribution estimation, which differ in the exact problem they target and how they handle the associated uncertainty.

Bayesian inference

One of the central approaches to perform parameter inference is Bayesian inference [85]. In Bayesian inference, the likelihood, together with a prior distribution over parameters $p(\boldsymbol{\theta})$ is used to compute the *posterior distribution* $p(\boldsymbol{\theta} \mid \boldsymbol{x})$ for a given observation $\boldsymbol{x}_o$ via Bayes theorem

$$p(\boldsymbol{\theta} \mid \boldsymbol{x}_o) = \frac{p(\boldsymbol{x}_o \mid \boldsymbol{\theta})\, p(\boldsymbol{\theta})}{\int p(\boldsymbol{x}_o \mid \boldsymbol{\theta})\, p(\boldsymbol{\theta})\, d\boldsymbol{\theta}} = \frac{p(\boldsymbol{x}_o \mid \boldsymbol{\theta})\, p(\boldsymbol{\theta})}{p(\boldsymbol{x}_o)}. \tag{2.18}$$

The prior distribution $p(\boldsymbol{\theta})$ encodes available knowledge about the parameters. The normalizer $p(\boldsymbol{x}_o)$ is known as the Bayesian evidence, or marginal likelihood. Bayes' theorem can also handle multiple observations that are independent given $\boldsymbol{\theta}$, in which case the joint likelihood factorizes. Thus, Bayesian inference can be used to compute the posterior distribution given one or more observations.

A classic introductory example to Bayesian inference is estimating the probability of a coin landing on heads, denoted by θ_c. After specifying a prior distribution over θ_c, for example the uniform distribution over the unit interval, the posterior can be analytically computed for one or more independent coin flips (Fig. 2.2). As more coin flips are incorporated, the posterior distribution will converge to the true probability of landing heads.

In general, however, it is rarely possible to compute the posterior analytically. This is because the integral of the Bayesian evidence in Eq. (2.18) can be challenging to estimate, especially in high dimensions. Many methods like Markov chain Monte Carlo (MCMC), variational inference, and sequential Monte Carlo have been developed to approximate the posterior in such cases [85–87].

Simulation-based inference Simulation-based (Bayesian) inference (SBI) is a flexible and general approach to performing Bayesian (parameter) inference with complex simulators that have potentially intractable likelihoods, where classical methods like MCMC are not applicable [41]. Here, I focus on *neural* SBI, which involves training a deep probabilistic model using simulations. Specifically, as before, let $s : \boldsymbol{\theta} \mapsto \boldsymbol{x}$ be a simulator that takes parameters $\boldsymbol{\theta}$ as input and outputs a simulation $\boldsymbol{x}$. There are no additional requirements for the simulator, in particular, it need not be differentiable and the associated likelihood $p(\boldsymbol{x} \mid \boldsymbol{\theta})$ may only be implicitly defined. The goal of SBI is to obtain an estimate of the Bayesian posterior distribution $p(\boldsymbol{\theta} \mid \boldsymbol{x}_o)$ (Eq. (2.18)) only using simulations from s.

SBI works in two steps: First, after specifying a prior distribution over the parameters $p(\boldsymbol{\theta})$, samples from the joint distribution $p(\boldsymbol{x}, \boldsymbol{\theta})$ are created by sampling from the prior and then simulating with s. Next, the resulting dataset $\mathcal{D} = \{\boldsymbol{\theta}_i, \boldsymbol{x}_i\}_{i=1}^n$, consisting of n parameter-simulation pairs, is used to train a deep probabilistic model within one of the various different SBI approaches. Neural posterior estimation (NPE) directly targets the posterior distribution by training a conditional deep generative model, such as a normalizing flow or diffusion model (Sec. 2.1.2), to approximate $q_\phi(\boldsymbol{\theta} \mid \boldsymbol{x}) \approx p(\boldsymbol{\theta} \mid \boldsymbol{x})$ [42, 88]. In neural likelihood estimation, a surrogate likelihood $q_\phi(\boldsymbol{x} \mid \boldsymbol{\theta}) \approx p(\boldsymbol{x} \mid \boldsymbol{\theta})$ is trained, typically using a normalizing flow. In turn, the tractable density of the surrogate likelihood allows the use of classical methods for Bayesian inference, such as MCMC [43, 89]. Similarly, neural ratio estimation uses deep classifiers (Sec. 2.1.1) to approximate the likelihood-to-evidence ratio, which again allows MCMC to be used to obtain the desired posteriors [90, 91]. Recently, many approaches have taken advantage of the flexibility of diffusion models to capture the full joint distribution or handle multiple observations [44, 88, 92]. Crucially, all of these approaches provide amortized inference after training. That is, they can be reused to obtain the posterior distribution for different observations without the need for retraining or additional simulations, which presents a significant advantage over more classical inference methods, such as MCMC or approximate Bayesian computation (ABC, 93).

One of the central challenges of SBI is simulation efficiency, that is, to accurately estimate the posteriors using as few simulations as possible [41, 94]. This is because, in practice, simulators can be computationally expensive possibly requiring minutes or hours per simulation, which hinders users from creating large datasets of parameter-simulation pairs. Therefore, beyond amortized SBI, much research has gone into so-called sequential approaches [89, 91, 95–97]. In these approaches, for a fixed observation $\boldsymbol{x}_o$, different active learning schemes are used to query the simulator for relevant simulations. This process iteratively refines the posterior estimate. While these approaches sacrifice amortization, they are usually more simulation-efficient, often requiring substantially fewer simulations to obtain accurate posterior estimates.

Finally, it is worth noting that the prior-data fitted networks (PFNs) introduced in Sec. 2.1.3

can also be considered as an instance of amortized SBI. In this case, the prior is not defined over simulator parameters, but rather over probability distributions and the "simulator" corresponds to taking samples from these distributions. After training, PFNs compute the posterior predictive distribution given a dataset and a test point (Eq. (2.17)). Hence, unlike SBI, PFNs do not target the posterior directly. Nevertheless, the key feature of PFNs is their ability to provide *amortized Bayesian inference* for this posterior predictive distribution, enabling it to be used for training-free, in-context regression and classification.

Source distribution estimation

Here, we consider a less known approach to parameter inference, source distribution estimation, which differs from, yet is closely related to, Bayesian inference [45, 98]. Suppose one has a dataset of observations $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ each of which is assumed to be generated from an unobserved parameter $\boldsymbol{\theta}_i$. The parameters $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n$ are further assumed to be drawn independently from a shared distribution $\pi(\boldsymbol{\theta})$, often referred to as the source distribution. That is, similar to simulation-based inference, we assume a generative model of the form:

$$\boldsymbol{\theta}_i \sim \pi, \quad \boldsymbol{x}_i \sim p(\boldsymbol{x} \mid \boldsymbol{\theta}_i). \tag{2.19}$$

The task of *source distribution estimation* is to estimate the unknown source distribution π from the dataset of marginal observations $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ alone. Specifically, for a parameterized source distributions q_ϕ, we aim to find parameters ϕ such that

$$p_o(\boldsymbol{x}) \approx q_\phi^{\#}(\boldsymbol{x}) = \int p(\boldsymbol{x} \mid \boldsymbol{\theta})\, q_\phi(\boldsymbol{\theta})\, d\boldsymbol{\theta}, \tag{2.20}$$

where $p_o(\boldsymbol{x})$ is the distribution of observations (i.e. $\boldsymbol{x}_i \sim p_o(\boldsymbol{x})$ for each observation), and $q_\phi^{\#}(\boldsymbol{x})$ denotes the pushforward distribution of the estimated source with respect to the likelihood. In a hierarchal Bayesian setting, estimating of a shared source distribution for all observations corresponds to estimating the prior distribution from data. In this case, source distribution estimation is also known as empirical Bayes [99, 100]. These data-based priors can capture shared structure across samples and offer a systematic approach to the often difficult task of manually specifying a prior distribution [101]. However, source distribution estimation is an important inference problem in its own right, in cases where statements about individual parameters and observations are not required.

Unlike Bayesian inference, source distribution estimation is inherently ill-posed: many distinct sources can induce the same marginal distribution $p(\boldsymbol{x})$ given a sufficiently complex likelihood. As an example, we consider a variation of the classic coin-flip experiment. Instead of repeatedly flipping the same coin to estimate its (posterior) probability for heads θ_c, we flip a large number of different, independent coins exactly once. Source distribution estimation now asks: What distribution of probabilities $\pi(\theta_c)$ results in the observed fraction of heads? This question does not have a unique answer. Suppose we have observed 75% heads. Then, one possibility is that all coins are identical, each with a probability of $\theta_c = 0.75$. Formally this would correspond to the point mass $\pi(\theta_c) = \delta_{0.75}(\theta_c)$. Alternatively, half of the coins could be fair, while the other half would always land on heads, formally $\pi(\theta_c) = 0.5\delta_{0.5}(\theta_c) + 0.5\delta_{1.0}(\theta_c)$. On average, we would still observe 75% heads. $\pi(\theta_c)$ could even be continuous, for example, a uniform distribution centered around 0.75. More precisely, any distributions on the unit interval $[0, 1]$ with an expected value of 0.75 is a valid source distribution in this case (see Fig. 2.2b with additional examples). Therefore, additional

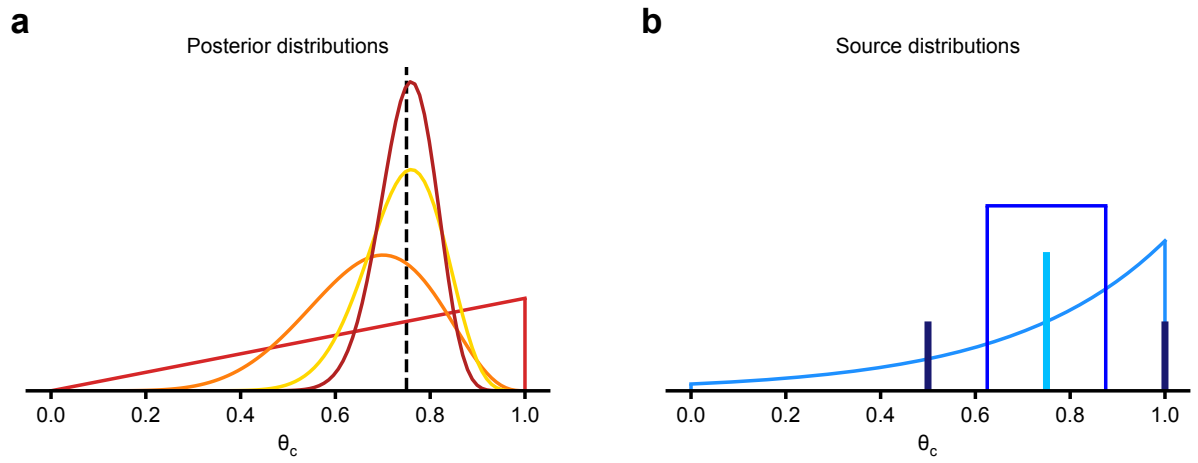


Figure 2.2: Bayesian posterior distributions vs. source distributions. (a) Example posterior distributions for the coin flip example. Each posterior is uniquely determined by the prior for the probability of heads θ_c and the one or more independent observations. In this simple case, the posteriors shown correspond to 1, 10, 25, and 50 throws of the coin and are all instances of the Beta distribution $\mathrm{Beta}(1+\#\text{tails}, 1+\#\text{heads})$. With an increasing number of throws, the posterior concentrates around the ground truth parameter, which is indicated by the dashed line. **(b)** Example source distributions for the coin flip example. Unlike the Bayesian posterior distributions, the source distributions in the coin flip example are not unique. That is, all four different distributions result in 75% heads observed, and are thus valid solutions to the source distribution estimation problem. Thick lines indicate point masses. The two continuous distributions are a uniform distribution and a truncated exponential distribution.

constraints on the estimated source distribution $q_\phi(\boldsymbol{\theta})$ are necessary to resolve the ill-posedness of source distribution estimation.

Simulation-based source distribution estimation Although the machine learning community has focused more on simulation-based (Bayesian) inference than on source distribution estimation, recent work has considered the estimation of increasingly complex, high-dimensional source distributions using deep probabilistic models [45, 46, 102]. Such approaches often include an SBI-like component, where either a surrogate likelihood or a direct posterior estimator is trained using a dataset of simulations. These trained deep probabilistic models can then be used to estimate source distributions using various methods, including adversarial learning, maximum likelihood estimation or expectation maximization. Beyond learning data informed priors in the context of empirical Bayes, these approaches have been used in a variety of applications under different names, ranging from “unfolding” in particle physics to “population inference” in cell biology [47, 103, 104].

2.2 Deep neural networks for physiological time series

The emergence of deep learning has made the creation of neural architectures that can handle sequential data, or time series, an active field of study. This section provides an overview of the neural architectures typically used to extract information from, or construct representations of, time series.

In this thesis, I focus solely on regular time series, in which individual data points are equally spaced in time. In other words, the time period between every pair of subsequent data points, x_t and x_{t+1}, is constant. The inverse of this period is the recording frequency of the time series and is measured in hertz (Hz). Additionally, time series can be multivariate, that is, each data point can be a vector $\boldsymbol{x}_t \in R^C$, and the C different dimensions are referred to as channels. Together with the length T, that is, the number of data points $\boldsymbol{x}_t$, $1 \leq t \leq L$, a time series is a matrix $\boldsymbol{x} \in \mathbb{R}^{C \times L}$. Only in the special case of univariate time series with only one channel, times series $\boldsymbol{x} \in \mathbb{R}^L$ can be represented as vectors, and its data points $x_t \in \mathbb{R}$ as scalars.

2.2.1 A closer look at deep learning

Deep learning uses deep neural networks to automatically learn representations of and from data for tasks such as prediction, generation, and inference [8]. In Sec. 2.1, I discussed how these learning problems are typically framed from a probabilistic perspective. However, I glossed over the construction of the parameterized functions f_ϕ, which underlie the explicitly or implicitly parameterized distributions q_ϕ.

In deep learning, these functions are deep neural networks. These networks are simply compositions of large numbers of differentiable operations, some of which are nonlinear. For example, consider

$$f_\phi(\boldsymbol{x}) = f^{(L)} \circ f^{(L-1)} \circ \cdots \circ f^{(1)}(\boldsymbol{x}) = \phi_L(W_L\, \phi_{L-1}(W_{L-1} \cdots \phi_1(W_1 \boldsymbol{x} + \boldsymbol{b}_1) \cdots + \boldsymbol{b}_{L-1}) + \boldsymbol{b}_L), \quad (2.21)$$

where the repeating subgroups of operations, such as f_ϕ^i, are typically referred to as layers in a network. Here, Eq. (2.21) is a composition of matrix multiplication with weight matrices, W_i, and vector addition with bias vectors $\boldsymbol{b}_i$ followed by a pointwise nonlinear activation function, ϕ_i. Therefore, it is an example of an (L-layer) multi-layer perceptron, which is one of the oldest types of deep neural networks [105].

Since all operations within a neural network are differentiable, their composition is also differentiable. This enables the network's parameters to be optimized using (stochastic) gradient descent with respect to a loss computed on the training data. Examples of such loss functions were introduced in Eq. (2.6) and Eq. (2.14). Modern deep learning frameworks such as PyTorch [106] compute the gradients required for optimization using reverse-mode autodifferentiation, or backpropagation [107]. Stochastic gradient descent is then typically performed using specialized optimizers, most famously Adam [108, 109], to find good parameters ϕ. The type of data a neural architecture is supposed to process greatly affects its design. As we will see, there are many choices for time series data alone, each with its own strengths and weaknesses.

2.2.2 Time series architectures

Here, I discuss several neural architectures that have been specifically designed to process and extract information from sequential data, or time series. Importantly, our discussion is limited to the

time series layers within these architectures that are crucial for capturing temporal dependencies. Neural architectures for time series typically have many layers or components that are not specific to time and are also used in many other neural architectures, for instance those tailored to images. Examples of these layers or components are normalization layers, activation functions, pooling layers and residual connections, to name a few [110–114].

Additionally, since the focus is on regular time series, I will limit the discussion to architectures for regular time series. A discussion of the transformer architecture and attention layers [75], as well as similar set-based architectures is also beyond the scope of this thesis, despite these architectures often being used to process both regular and irregular time series [115, 116].

Recurrent neural networks

Recurrent neural networks (RNNs) are among the earliest and most popular deep architectures for processing sequential data [9]. They work by maintaining hidden states, $\boldsymbol{h}_t$, that evolve over time. A classical recurrent layer is defined as

$$\boldsymbol{h}_t = \sigma\big(W_h \boldsymbol{h}_{t-1} + W_x \boldsymbol{x}_t + b\big), \tag{2.22}$$

where $\boldsymbol{x}_t$ is the input at time t, $\boldsymbol{h}_t, \boldsymbol{h}_{t-1}$ are the current and previous hidden states, σ is a nonlinear activation function, and W_h, W_x, $\boldsymbol{b}$ are learnable weight matrices and vectors. An RNN is then the composition of several such recurrent layers.

In theory, RNNs can capture arbitrary dependencies between time points [117]. However, their use has been limited by their challenging training dynamics. Specifically, training RNNs requires backpropagation through time, which is computationally expensive and can result in vanishing or exploding gradients [118]. RNN variants, such as long short-term memory (LSTM, 10) and gated recurrent units (GRUs, 119), were developed to address these issues. Nevertheless, applying them to large-scale problems with long time series remains challenging. Despite these limitations, RNNs are still routinely used within deep probabilistic models for countless applications, ranging from time-series classification and imputation to time-series generation and the extraction of latent dynamics [24, 120–122].

Temporal convolutional neural networks

An alternative to RNNs are temporal convolutional neural networks (CNNs). Inspired by the development of 2D convolutions for processing images [12, 51], temporal CNNs process time series by convolving parameterized convolution kernels along the time dimension. Assuming a univariate time series $\boldsymbol{x}$ as input, a convolutional layer convolves the kernel $\boldsymbol{k} \in \mathbb{R}^{2M+1}$ across the time series channels with

$$h_t = \sigma\left(\sum_{m=-M}^{M} k_m \cdot x_{t-m} + b\right), \tag{2.23}$$

where the kernel $\boldsymbol{k}$ is indexed from $-M$ to M, b is a scalar bias, and σ is again a nonlinear activation function. For multivariate time series, individual convolutions are summed over different time series channels.

Similar to RNNs, a temporal CNN is then the composition of several such convolutional layers, often together with pooling layers to aggregate information. Unlike RNNs, however, temporal CNNs do not suffer from challenging training dynamics and are easier to use in practice. The main limitation of temporal CNNs is that they cannot capture temporal dependencies beyond their

receptive field. The size of the receptive field is given by the number of time series elements that can influence each other through repeated applications of convolution kernels across different layers. Therefore, temporal CNNs are especially useful for extracting localized features from a time series and are typically used in applications such as time-series classification [17, 123–125]. Capturing long-range dependencies with temporal CNNs is challenging as naively scaling up the kernel dimensionalities results in networks that are hard to train. For some applications, tailored architectures can be used to overcome this issue. For example, the WaveNet architecture uses specially constructed convolution kernels with receptive fields spanning several thousand time series elements to generate high-fidelity audio [126].

Structured state space models and structured convolutions

Recently, deep structured state space models have emerged as a powerful alternative to classical RNNs and temporal CNNs [13, 14, 127]. State space models were originally developed to overcome the quadratic time complexity of self-attention in transformers while still being able to capture very long-range dependencies [128].

Deep structured state space models are, as the name suggests, based on the (discretized) state space equations. Let

$$\begin{aligned} \boldsymbol{s}_t &= A\boldsymbol{s}_{t-1} + Bx_t \\ h_t &= C\boldsymbol{s}_t + Dx_t, \end{aligned} \tag{2.24}$$

where $\boldsymbol{s}_t \in \mathbb{R}^d$ is the hidden state, and x_t and h_t are the (univariate) input and output at time t. The system matrices $A \in \mathbb{R}^{d\times d}, B \in \mathbb{R}^{1\times d}, C \in \mathbb{R}^{d\times 1}$, and $D \in \mathbb{R}^{1\times 1}$ contain the trainable parameters and d is a hyperparameter. Because of their recurrent nature, state space models are closely related to the previously introduced RNNs. The crucial difference, however, is that the recurrence in Eq. (2.24) is linear.

In S4 [13], the first version of a deep structured state space model, the system matrices are carefully constructed and initialized to enable capturing long-range dependencies and avoiding back-propagation through time. One key idea behind this careful construction is the equivalence between linear recurrence and convolution with a implicit, but highly structured kernel. Formally, this kernel is given by

$$\boldsymbol{k} = (CB,\ CA^1B,\ CA^2B,\ \dots CA^{L-1}B)^T \in \mathbb{R}^L. \tag{2.25}$$

Hence, the output of the state space equations can be written as a one-dimensional convolution (Eq. (2.23)) between the input time series $\boldsymbol{x}$ and the implicitly defined kernel $\boldsymbol{k}$, which is constructed by iterated application of the system matrices. Together with the careful parametrization of the system matrices [129], these implicit kernels have exponentially decaying weights that can capture very long range dependencies (Fig. 2.3a). Additionally, representations of these kernels in frequency domain can be used to efficiently compute these convolutions during training.

Although deep structured state space models are theoretically less expressive than RNNs, their stable training dynamics and ability to capture long-range dependencies spanning several thousand timesteps have led to their widespread adoption. As such, state space models have found applications in many domains, including language modeling, audio generation, and imputation [20, 130, 131].

Furthermore, the development of these recurrent state space models has informed the creation of modern variants of temporal CNNs that place a strong inductive bias on their kernels. This

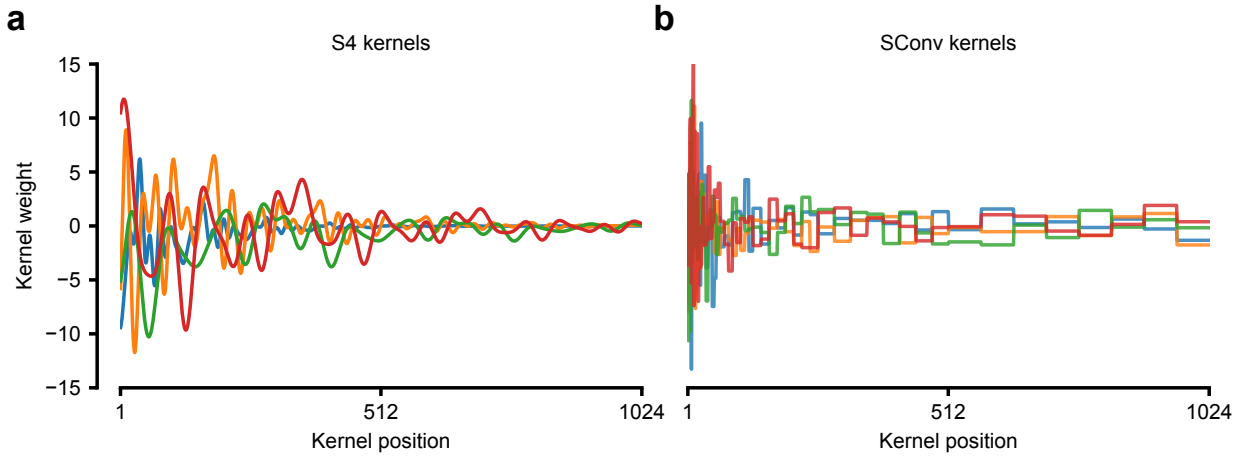


Figure 2.3: Convolution kernels of structured state space models and structured convolutions. (a) Four convolution kernels obtained by unrolling the state space matrices of an S4 layer over 1024 timesteps. The kernel weights decay exponentially as a function of the kernel position. **(b)** Four convolution kernels created via structured convolutions (SConv), where standard convolution kernels are concatenated with increasingly upscaled receptive fields and downscaled weights. For these examples, nearest-neighbor upsampling was used, and the constant sections (especially visible towards the right) are represented by a single scalar parameter.

inductive bias enables these CNNs to handle much larger kernel sizes than was previously possible [132, 133]. One example of such a modern variant are structured convolutions [133], where long convolution kernels are created by concatenating several standard convolution kernels to mimic the implicit kernels in structured state space models. Formally, consider a set of convolution kernels $S = \{\boldsymbol{k}_i\}_{i=1}^{K}$, each with kernel weights $\boldsymbol{k}_i \in \mathbb{R}^d$. A structured convolution kernel is then constructed via

$$\boldsymbol{k}_{struct} = \mathrm{Cat}(S) = \frac{1}{Z}\big[\tilde{\boldsymbol{k}}_0, \tilde{\boldsymbol{k}}_1, \ldots, \tilde{\boldsymbol{k}}_{N-1}\big], \quad \tilde{\boldsymbol{k}}_i = \alpha^i \,\mathrm{Upsample}_{2^{i-1}d}(\boldsymbol{k}_i) \quad \text{for } i > 0, \tag{2.26}$$

where $0 < \alpha < 1$ is a scaling factor. Therefore, subsequent kernel weights are progressively downscaled and their receptive fields are upscaled via simple interpolation. This procedure mimics the exponentially decaying structure of the implicit kernels in structured state space models (Fig. 2.3b). Importantly, the number of parameters increases only logarithmically with sequence length. Moreover, efficient training is possible by using fast Fourier transforms to shift the computation to frequency space.

2.2.3 The persistence of human-engineered features

Although this chapter has focused primarily on deep learning and neural architectures for time series—which aim to automate the process of feature extraction—human-engineered time-series features are still commonly used in applications related to physiological time series. Typical time series features include the mean, variance, higher moments, autocorrelation, spectral characteristics, and extreme points [134]. More importantly, depending on the application, specialized features can be constructed to encode domain knowledge. For example, the number of spikes is a highly informative statistic when working with recordings of neural dynamics, but would make little sense in other contexts [135].

The ability to encode domain knowledge is one major reason why human-engineered features are still used despite existing deep learning approaches and architectures. Extracting features of a similar quality with deep learning often requires a lot of training data [29]. Therefore, especially in data-sparse regimes, human-engineered features can, in some cases, still provide a powerful alternative. Another reason for the persistence of human-engineered features is that they can provide increased robustness. Often, deep neural networks are sensitive to slight changes in the data [33]. Specifically, if the time series used for testing or inference comes from a slightly shifted distribution than that used for training, the network's output can vary wildly and become unpredictable. Increasing the robustness of deep neural networks to these shifts is a major challenge and an active area of research [136, 137]. Human-engineered features, such as the number of spikes, can be less sensitive to slight shifts or corruptions in the data because they are typically implemented using a relatively simple operations.

Chapter 3

Publications

3.1 Overview

In this chapter, I provide an overview of the four publications that form the basis of this thesis. Each section introduces one publication and includes a discussion of the project's motivation, as well as an overview of methodological aspects and experimental results. The full papers are included in the appendix.

1. Julius Vetter, Kathleen Lim, Tjeerd M.H. Dijkstra, Peter A. Dargaville, Oliver Kohlbacher, Jakob H. Macke, and Christian F. Poets. *Neonatal apnea and hypopnea prediction in infants with Robin sequence with neural additive models for time series.* PLOS Digital Health 3, no. 12 (2024).

2. Julius Vetter, Jakob H. Macke*, and Richard Gao*. *Generating realistic neurophysiological time series with denoising diffusion probabilistic models.* Patterns 5, no. 9 (2024).

3. Julius Vetter*, Guy Moss*, Cornelius Schröder, Richard Gao, and Jakob H. Macke. *Sourcerer: Sample-based maximum entropy source distribution estimation.* Advances in Neural Information Processing Systems 37, (2024).

4. Julius Vetter*, Manuel Gloeckler*, Daniel Gedon, and Jakob H. Macke. *Effortless, simulation-efficient Bayesian inference using tabular foundation models.* Advances in Neural Information Processing Systems 38, (2025).

* denotes equal contribution.

3.2 Neonatal apnea and hypopnea prediction in infants with Robin sequence with neural additive models for time series

3.2.1 Motivation

Neonatal apnea and hypopnea, that is, repeated cessations in breathing or shallow breathing for several seconds during sleep, can negatively impact healthy infant development. Infants suffering from neonatal apnea and hypopnea require intensive treatment and supervision, often in a neonatal intensive care unit. Treatment may include labor-intensive manual stimulation by skilled personnel who monitor polysomnography recordings in order to administer these interventions. Automating the process of apnea detection and prediction would allow for automatic stimulation, such as a movable mattress, to be triggered, thus reducing the strain on clinical practitioners [34].

Most existing work in this direction focused on the detection and prediction of apneas and hypopneas using approaches based on classical feature engineering [35–37, 138, 139]. The goal of this project was to use more powerful deep classifiers to predict upcoming neonatal apneas directly from available polysomnography recordings. However, while the previous feature-engineering-based approaches may not be as expressive or flexible as deep learning-based ones, they are generally more interpretable because they use comprehensible features and simple, possibly linear prediction methods. This interpretability is crucial for applying machine learning in high-stakes settings, such as clinical practice, because it allows practitioners to trust or distrust the model in use [30–32]. Therefore, another goal of this project was to ensure that the deep classifier is inherently interpretable and can thus provide insight into how it reached a given prediction.

3.2.2 Methods

We base our predictions of upcoming apneas and hypopneas on polysomnography recordings, which are a combination of several physiological time series that are routinely collected during sleep. In our case, these recordings include six different signal modalities: nasal pressure, abdominal and thoracic effort, heart rate (derived from an electrocardiogram), a photoplethysmogram, and two blood-related modalities, the SpO_2 and PCO_2 levels. These time series are measured at various frequencies ranging from 1 to 200 Hz.

Predicting a discrete set of events from time series is typically framed as a classification problem. In our case, we consider 30-second segments of polysomnography recordings with the aim of classifying them as normal breathing or pre-apnea/pre-hypopnea breathing. Given an overnight polysomnography recording with human-based annotations, we extract 30-second segments with their associated labels to train and test our a classifiers.

To create an expressive yet interpretable deep classifier (Sec. 2.1.1), we structure it as a generalized additive model (GAM), which additively aggregates information from the six different signal modalities. More specifically, we parameterize the probability of a time series segment $\boldsymbol{x}$ to be pre-apnea (or pre-hypopnea) breathing $p(y = \text{pre-apnea} \mid \boldsymbol{x})$ as

$$p(y = \text{pre-apnea / pre-hypopnea} \mid \boldsymbol{x}) = \sigma\left(\sum_{i=1}^{6} \alpha_i f_{\phi_i}(\boldsymbol{x}_i) + \beta\right), \tag{3.1}$$

where each additive contribution $\alpha_i f_{\phi_i}$ is the product of a scalar scaling factor α and a temporal CNN f_{ϕ_i} (Sec. 2.2.2), which directly operates on the physiological time series of the corresponding signal modality (Fig. 3.1a). These combinations of deep neural networks with GAMs are known as

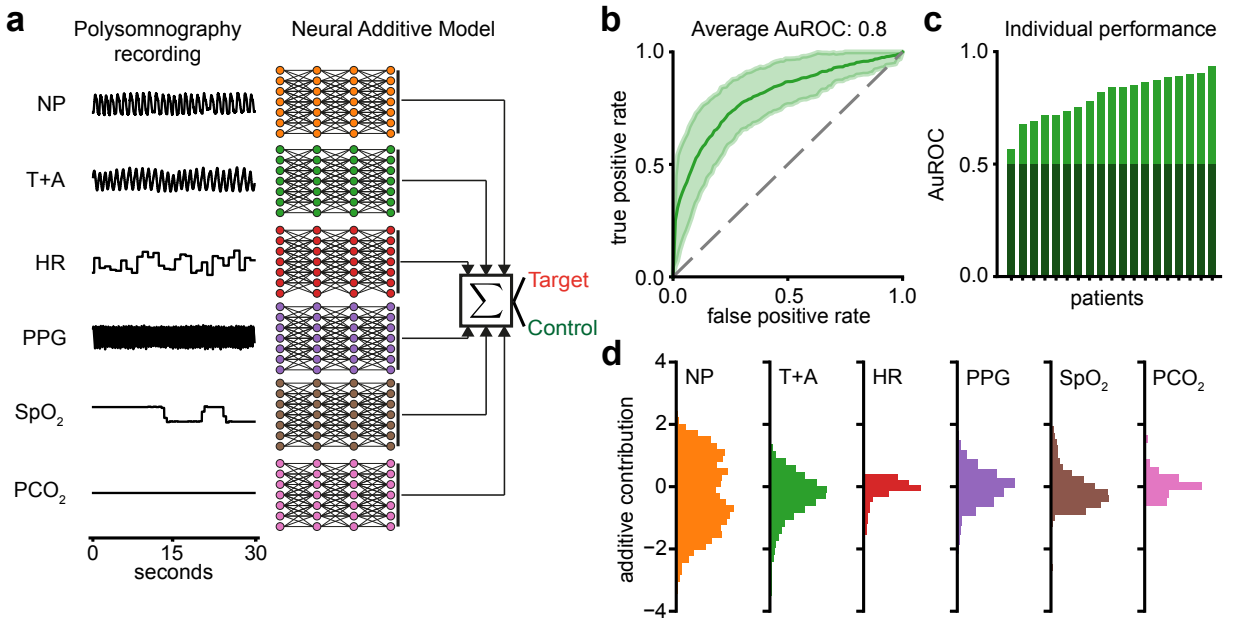


Figure 3.1: Neural additive models (NAMs) for interpretable prediction of neonatal apneas. (a) Example 30 second polysomnography time window used for prediction, which consists of the six signal modalities nasal pressure (NP), abdominal and thoracic effort (T+A), heart rate (HR), a photoplethysmogram (PPG), SpO_2 and PCO_2. In the NAM, each signal modality is passed trough temporal CNN and individual network outputs are summed to perform the prediction. **(b)** The area under the receiver operator characteristic curve (AuROC) averaged across patients. **(c)** AuROCs for the individual patients sorted by increasing performance. **(d)** Histograms of additive contributions over all patients and time windows. Larger positive or negative contributions have a bigger influence on the final prediction. Panels adapted from Vetter et al. [140].

neural additive models (NAMs, 141) and can be trained by maximizing the likelihood (Eq. (2.6)) using stochastic gradient descent.

Our NAM is interpretable in two ways: First, the additive contributions provide insight into which of the six signal modalities contributed the most to a given prediction. Large positive or negative additive contributions contribute more the logit in Eq. (3.1), and thus have more influence on the model's prediction. Second, class activation maps [142] associated with each temporal convolutional neural network (CNN) allow us to visualize the local time series features in each signal modality deemed predictive by the model. Thus, our time series NAM strikes a balance between a black-box deep neural network and the classical GAM setup, in which the additive model would be used together with scalar features.

3.2.3 Results

We applied our model to a dataset of 21 children with Robin syndrome, a condition that causes a large number of apneas and hypopneas [143]. The dataset was collected and annotated by our collaborators from the Tübingen University Hospital [144]. Using leave-one-out cross-validation, we compared the predictive performance of our model to various baselines, including methods that use classical feature engineering and black-box deep classifiers.

We demonstrate that our model effectively predicts upcoming apneas and hypopneas, as measured by the area under the receiver operating characteristic curve (AuROC, 145). Averaged across

patients, the model achieves an AuROC of 0.8 (Fig. 3.1b). However, prediction performance varies greatly between patients, with some achieving AuROCs higher than 0.9, and one close to a random performance of 0.5 (Fig. 3.1c). Overall, our model is comparable in performance to the baseline black box model and superior to baseline feature-engineering approaches (for the features considered).

With regards to model interpretability, we conduct a signal modality importance analysis to determine which of the six signals are, on average, most important for the model prediction. This is done by analyzing the additive contribution of individual modalities to the overall prediction across different patients. Modalities with many large positive or negative contributions have a stronger impact on the final prediction. As expected by our clinical collaborators, the analysis of these additive contributions shows that breathing-related modalities, particularly nasal pressure, were most important (Fig. 3.1d). Analyzing class activation maps showed, that within these breathing related modalities, the model focused on irregular or arrhythmic patterns. To a lesser extent, the model made use of information in heart rate or SpO_2 levels, exploiting abnormal levels in relation to previous events.

While these insights may not be surprising from a clinical perspective, they are nonetheless valuable because they enable practitioners to verify that the model uses clinically relevant features and does not rely on spurious shortcuts that could be ineffective in a real interventional setting, thus building trust. Overall, this paper presents another step toward automating the prediction of neonatal apnea and hypopnea. It demonstrates that expressive deep classifiers can be used directly on polysomnography recordings while maintaining interpretability. However, more studies applying these models in the considerably more challenging interventional setting are needed before they can be used in clinical practice.

3.2.4 Contributions

Contributions following CRediT:

Julius Vetter Conceptualization, Formal analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review & editing
Kathleen Lim Data curation, Writing – review & editing
Tjeerd M. H. Dijkstra Supervision, Writing – review & editing
Peter A. Dargaville Supervision, Writing – review & editing
Oliver Kohlbacher Funding acquisition, Supervision, Writing – review & editing
Jakob H. Macke Conceptualization, Funding acquisition, Supervision, Writing – review & editing
Christian F. Poets Conceptualization, Data curation, Funding acquisition, Supervision, Writing – review & editing

3.3 Generating realistic neurophysiological time series with denoising diffusion probabilistic models

3.3.1 Motivation

Neuroscience is full of densely sampled and multivariate neurophysiological timeseries. Classical examples include electroencephalography (EEG) and local field potentials (LFP), where often more than 100 time series channels are simultaneously recorded at a high temporal resolution. Recent diffusion models (Sec. 2.1.2) have been used to model shorter, multivariate time series, which allows tackling common time-series related tasks, such as imputation [19] and forecasting [39, 40, 146]. Additionally, diffusion models have been to applied to high-fidelity audio generation, which requires modeling a densely sampled, univariate time series [147]. However, the ability of diffusion models to capture densely sampled, multivariate neurophysiological timeseries had not yet been explored. Thus, we began this project by asking the following question: *Can diffusion models also be used to accurately model neurophysiological time series?*

Access to a generative model that accurately captures the underlying distribution of neurophysiological time series would enable various neuroscientific applications, such as creating synthetic datasets for simulators, imputing missing signal channels, classifying underlying brain states, and detecting outliers in hours of recordings. Crucially, the training of such a model must be data-efficient because many neurophysiological time series datasets contain only a few recordings.

3.3.2 Methods

In order to accurately model densely sampled, high-dimensional time series with diffusion models, we develop an expressive diffusion backbone based on structured convolutions (Sec. 2.2.2). These convolutions, which have kernels across multiple length scales, allow DDPMs to handle time series with thousands of time points and capture the multiscale nature of neural dynamics. Our architecture uses layers of structured convolutions interleaved with time-point-wise linear layers, normalization layers, and GELU nonlinearities [111]. This construction follows the classical transformer architecture [75].

Beyond constructing a tailored diffusion backbone, we explore the possibility of incorporating neuroscientific domain knowledge about power spectra directly into the generative process (Fig. 3.2a). Specifically, many neurophysiological time series exhibit a $1/f$-like power law in the frequency domain. To incorporate this power-law structure into the diffusion process, we replace the isotropic Gaussian noise typically used in DDPMs with "colored" noise from an Ornstein-Uhlenbeck (OU) process. This modification changes the loss in Eq. (2.14) to

$$\mathcal{L}(\phi) = \mathbb{E}_{\mathbf{x}_0,\boldsymbol{\epsilon},t}\left[(\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\phi(\mathbf{x}_t, t))^\top \boldsymbol{\Sigma}_\rho^{-1} (\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\phi(\mathbf{x}_t, t))\right], \tag{3.2}$$

where again $\boldsymbol{x}_t = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}$, and $t \sim U(0, T)$, but this time $\epsilon \sim \mathcal{N}(0, \Sigma_\rho)$ is colored noise from a OU process with covariance matrix Σ_ρ. The lengthscale ρ of the OU process Σ_ρ is a hyperparameter. Notably, the operations including the precision matrix Σ^{-1} of the OU process can still be efficiently computed even for long sequences thanks to its tridiagonal structure.

3.3.3 Results

To evaluate data-efficiency and generation quality of our DDPM, we apply it to several datasets containing various types of neurophysiological time series, including EEG, LFP, and electrocorticog-

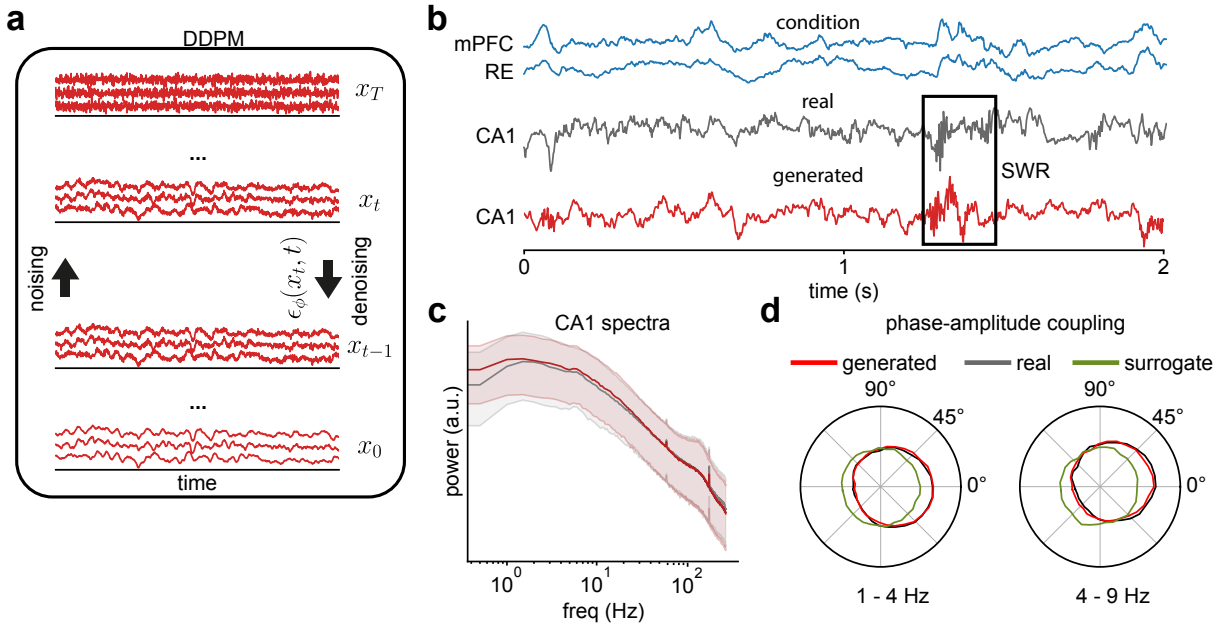


Figure 3.2: Denoising diffusion probabilistic models (DDPMs) for neurophysiological time series. **(a)** A DDPM is trained to gradually denoise white noise or, in our case, noise sampled from an Ornstein-Uhlenbeck process, transforming it into samples from the underlying distribution of a neurophysiological time series. **(b)** Example of conditional generation or imputation in a three channel LFP recorded from the rat. Conditioned on the mPFC and RE channels, the DDPM generates an imputation of the CA1, which exhibits characteristic sharp wave ripples. **(c)** Distribution of power spectra (median and 10% / 90% percentiles) of real versus generated time series. **(d)** Phase amplitude coupling between the CA1 and mPFC channel for the real, generated, and surrogate time series as well as different driver frequencies. Panels adapted from Vetter et al. [148].

raphy (ECoG) recordings. In our analysis of the generated time series, we focus on the statistics that are of interest to neuroscientists. These statistics include power spectra, phase-amplitude coupling, and downstream performance in neural decoding tasks or brain state classification.

First, we consider a dataset of three-electrode LFPs from rats recorded at 600 Hz in the medial prefrontal cortex (mPFC), hippocampal CA1 region, and thalamic nucleus reuniens (RE) [149]. After training, the diffusion model can be used for both conditional and unconditional generation of LFPs. The generated time series are visually similar to the real time series and exhibit fine-grained neuroscientific features, such as sharp wave ripples (Fig. 3.2b). Furthermore, neuroscientific statistics such as the distribution of power spectra of the CA1 channel and the phase-amplitude coupling between the CA1 and mPFC channels are accurately captured (Fig. 3.2c,d). As a baseline, we also consider surrogate LFPs obtained using spectral surrogates [150]. By construction, these surrogates have the same power spectra as the original time series. However, unlike the diffusion-generated time series, they do not exhibit higher-order structure across signal channels (Fig. 3.2d).

Next, we demonstrate how these models can be used to impute missing time series channels in a dataset of human ECOG recordings with over 100 time series channels [151]. Diffusion-based probabilistic imputation improves performance in a downstream neural decoding task and outperforms simpler, deterministic imputation methods.

Finally, we train a conditional diffusion model on ECoG recordings from a macaque monkey in different stages of wakefulness and anesthesia [152]. By evaluating the density using the prob-

ability flow equation [69], we find that our conditional diffusion models can act as generative classifiers (Eq. (2.9)): They can distinguish between anesthetized and awake brain states and detect outliers. Notably, many of the datasets used only contain a few hundred training samples, demonstrating the data efficiency of our DDPMs.

In summary, this paper shows that DDPMs with carefully designed, domain-specific modifications can be trained on a relatively small number of samples to generate high-quality, densely sampled, multivariate neurophysiological time series. Notably, across all datasets, the main improvement in performance over baseline generative models is achieved via the expressive diffusion backbone with structured convolutions. In contrast, the OU diffusion process only improves the quality of the generated time series for certain datasets. An important limitation of our DDPMs is that they can only model continuous neurophysiological recordings. In a follow-up project led by collaborators, we use similar diffusion backbones based on structured state space models, together with a tailored autoencoder, to model discrete neurophysiological time series of neural spiking activity [153].

3.3.4 Contributions

Contributions following CRediT:

Julius Vetter: Conceptualization, Formal analysis, Methodology, Visualization , Software, Writing – original draft, Writing – review & editing
Jakob H. Macke: Conceptualization, Supervision, Writing – review & editing, Funding acquisition
Richard Gao: Conceptualization, Supervision, Writing – review & editing

3.4 Sourcerer: Sample-based maximum entropy source distribution estimation

3.4.1 Motivation

As described in Sec. 2.1.4, estimating source distributions is an inherently ill-posed problem. Even with a reasonably complex simulator, an identical pushforward distribution can result from many different source distributions of simulator parameters. However, existing neural-based methods for estimating source distributions do not impose additional constraints, which leaves considerable uncertainty about the estimated distribution [45–47]. This uncertainty can prevent scientific interpretation and seriously affect downstream results (e.g., when Bayesian inference is performed using the estimated distribution as a prior).

One way to address this issue is to select a source distribution that avoids unnecessary and potentially implicit assumptions. In other words, we want to select the distribution that is maximally uncertain while still matching the constraint of being a proper source distribution. Using entropy as a measure of uncertainty leads to the principle of maximum entropy, which states that out of a family of feasible distributions, the one with the maximum entropy should be chosen [154, 155]. In this project, we investigate whether the maximum entropy principle can indeed be used to address the ill-posedness of source distribution estimation, from both theoretical and practical points of view.

3.4.2 Methods

First, we demonstrate that our application of the maximum entropy principle theoretically resolves the ill-posedness of the source distribution estimation problem. In other words, we show that the maximum entropy source distribution is unique. To prove this, we exploit a key property of source distributions: they are closed under averaging and thus form a convex set. Since (differential) entropy is a strictly concave functional on this set of distributions, the uniqueness of the maximum entropy source distribution follows, given some compactness assumptions (see Proposition 2.1 in the paper for details). In very simple cases, the maximum entropy source distribution can be derived analytically: For the coin flip example in Sec. 2.1.4, the truncated exponential distribution (Fig. 2.2b) is the unique maximum entropy source distribution.

For more challenging cases, our theoretical result suggests that one should solve the following constrained optimization problem:

$$\arg\max_{\phi} \quad H(q_\phi) \quad \text{s.t.} \quad D(q_\phi^{\#}, p_o) = 0, \tag{3.3}$$

where q_ϕ is the source distribution parameterized by a deep generative model (e.g. a normalizing flow) and $H(q_\phi)$ the entropy of the estimated source. $D(q_\phi^{\#}, p_o) = 0$ is an arbitrary distance metric that measures the distance between the distribution over observations p_o and the push-forward distribution of the estimated source $q_\phi^{\#}$. A distance of zero is equivalent to $p_o = q_\phi^{\#}$ and this constraint thus ensures that the estimated distribution is a valid source distribution (Eq. (2.20)).

In practice, however, solving this constrained optimization problem exactly is intractable because we only have access to samples from p_o and can therefore only estimate D. To circumvent this issue, we relax the above problem and instead target the regularized objective

$$\arg\max_{\phi} \quad \lambda H(q_\phi) - (1-\lambda)\log(D(q_\phi^{\#}, p_o)), \tag{3.4}$$

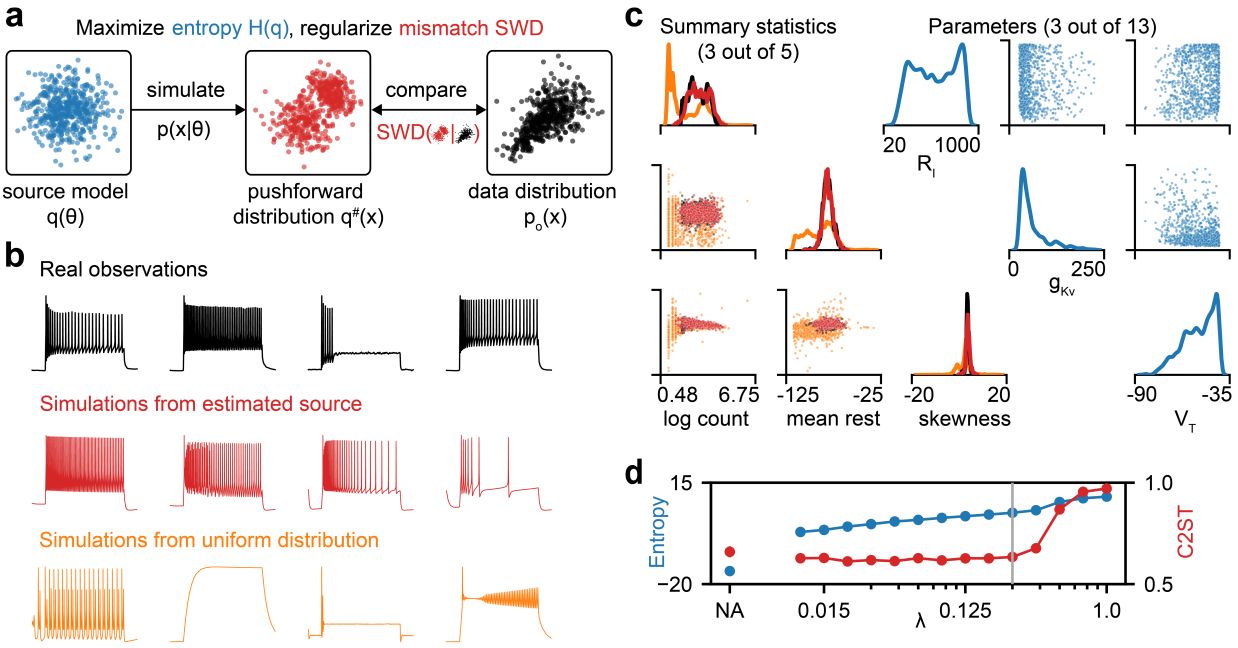


Figure 3.3: High-entropy source distribution estimation with Sourcerer. (a) Sourcerer works by computing the sliced-Wasserstein distance (SWD) between the dataset of observations and the dataset of simulations of parameterized source distributions. The mismatch is then used to optimized the parameters of the source distribution with gradient descent. Furthermore, the source distribution in parameter space is regularized towards having high entropy. This enables Sourcerer to explore the space of source distributions and to converge to one with high-entropy. We apply Sourcerer to a parameter inference task in neuroscience, where source distributions of Hodkgin-Huxley model parameters are estimated from a dataset of recordings of action potentials. **(b)** Example observations from the dataset together with simulations produced by sampling from the estimated source distributions and simulating. As a baseline, we also show simulations obtained with parameters sampled from a uniform distribution. **(c)** *Left:* Empirical distribution of observations in the space of summary statistics compared to distributions of simulations obtained from the estimated source (red) and uniform distribution (orange). *Right:* The estimated source distribution. In both cases, one- and two-dimensional marginals are shown. **(d)** C2ST accuracy, and (differential) entropy for various strengths of the regularization parameter λ, or without regularization (NA). Panels adapted from Vetter et al. [156].

which we maximize for some fixed regularization hyperparameter λ (Fig. 3.3a). Here, the equality constraint is approximated by measuring the distance between the distributions of simulations and observations using a sample based distance. In this project, we mainly consider the sliced-Wasserstein distance (SWD) [49, 157]. The SWD scales favorably with the number of observations as it is the average of many fast, one-dimensional projections of the Wasserstein distance, which can be computed in linearithmic time.

We typically use a normalizing flow (Sec. 2.1.2) to parameterize the source model, which enables us to estimate the differential entropy via the standard Monte Carlo approximation. However, our use of purely sampled based distances in Eq. (3.4) alternatively allows us to parameterize the source as a neural sampler. In a neural sampler, an arbitrary neural network (usually non-invertible, e.g., a MLP) transforms Gaussian samples into the desired distribution. Since the density of neural samplers is intractable, we estimate the entropy of the source using the Kozachenko-Leonenko estimator based on samples from the source [158].

3.4.3 Results

We compare our method, Sourcerer, to the state-of-the-art method from Vandegar et al. [45], which can learn complex source distributions but places no further constraints on them. Considering the same benchmark tasks as in Vandegar et al. [45], we demonstrate that Sourcerer achieves competitive or superior classifier-two-sample-test (C2ST, 159) accuracies while achieving substantially higher entropies in the estimated source distributions. Next, we demonstrate that our SWD-based approach can handle complex observations by applying Sourcerer to two high-dimensional simulators: the SIR and Lotka-Volterra benchmark tasks from Gloeckler et al. [160], which produce 50 and 100 dimensional observations, respectively. Finally, we apply our method to real action potential measurements and a single-compartment Hodgkin-Huxley model, using a set of well-established summary statistics [135, 161]. Again, Sourcerer achieves good agreement between real observations and simulations (Fig. 3.3b,c). Our results show that especially for this real-world application, regularizing the entropy of the estimated source is crucial. Without regularization, the agreement between observations and simulations is still good, but the estimated source has substantially lower entropy and misses many valid parameter regions (Fig. 3.3d). Overall, these results demonstrate that our method Sourcerer can estimate source distributions with substantially higher entropy than previous approaches, while maintaining the same level of agreement between simulations and observations.

3.4.4 Contributions

This publication was coauthored by me, Guy Moss, Cornelius Schröder, Richard Gao and Jakob H. Macke. I had the initial idea to target maximum entropy source distributions using a sample-based approach. Guy Moss and I together formalized this idea in terms of a constrained optimization problem, and proved the uniqueness of the maximum entropy source distribution. Guy Moss worked out the connection to the average posterior and proved they were not always valid source distributions. I performed the initial implementation of the code and the analysis pipeline for the synthetic tasks. Guy Moss and I implemented the application to the Hodgkin-Huxley model.

Guy Moss and I wrote the initial draft of the paper. Cornelius Schröder, Richard Gao and Jakob H. Macke provided feedback on the project throughout, and reviewed and edited the initial version of the paper.

3.5 Effortless, simulation-efficient Bayesian inference using tabular foundation models

3.5.1 Motivation

Simulation-based inference (Sec. 2.1.4) has become increasingly popular for performing Bayesian inference with intractable likelihoods. However, users face two challenges when applying SBI. First, since SBI methods are based on training deep probabilistic models to approximate the posterior distribution (directly in the case of NPE and indirectly in the case of NPE/NLE), users must be experienced enough to handle training and tuning such models. Second, if the simulator is expensive, accruing enough simulations to allow for accurate inference can be challenging. In this project, we asked ourselves whether we could address both issues—ease of use and the need for simulation efficiency—by repurposing tabular foundation models, specifically TabPFN [77, 80], for SBI. First, due to in-context learning, performing regression or classification with these foundation models is completely training-free, making them very easy to use. Second, in-context learning and prior-data-fitted networks (Sec. 2.1.3) offer a systematic approach to deal with limited data. They are pretrained on millions of small synthetic datasets, allowing them to handle the uncertainty associated by estimating the corresponding posterior predictive distribution. Therefore, issues such as overfitting, which complicate the training of deep probabilistic models from scratch with few simulations, can be avoided.

3.5.2 Methods

TabPFNv2 [80] is a deep probabilistic model that approximates univariate conditional densities $p(y \mid \boldsymbol{x}) \approx p(y \mid \boldsymbol{x}, \mathcal{D})$ via in-context learning using the context dataset $\mathcal{D} = \{y_i, \boldsymbol{x}_i\}_{i=1}^n$. To use TabPFN for parameter inference in SBI, we set $y = \theta$ and use the dataset of simulations and parameters as the context. However, simply setting $y = \theta$ would only allow us to model univariate posteriors. The key to estimating higher-dimensional posterior distributions with TabPFN is to use it autoregressively. After sampling the first parameter dimension, the samples are added to the context dataset, which is then extended to include the second parameter dimension. This adapted context then allows sampling of the second parameter dimension. This process is repeated until all $d_{\boldsymbol{\theta}}$ parameter dimensions are sampled (Fig. 3.4a). Formally, we have

$$p(\boldsymbol{\theta} \mid \boldsymbol{x}_o) \approx \prod_{j=1}^{d_\theta} q_\psi(\theta^j \mid \boldsymbol{\theta}^{<j}, \boldsymbol{x}_o, \mathcal{D}^{<j}), \tag{3.5}$$

where $\mathcal{D}^{<j} = \{\theta_i^j, [\boldsymbol{\theta}_i^{<j}, \boldsymbol{x}_i]\}$ is the extended dataset of parameter simulation pairs up to, but not including dimension j.

Although this autoregressive approach enables us to repurpose TabPFN for simulation-based inference, its effectiveness is limited by TabPFN's context size. As an in-context learner, TabPFN operates on entire datasets using self-attention layers, whose computational complexity scales quadratically with the number of samples in the dataset. Therefore, in practice, TabPFN effectively has a soft limit of around 10,000 context samples. After reaching this limit, computation becomes very expensive, and performance improvements diminish. Therefore, the available context should be used as efficiently as possible by filling it with relevant simulations for observation $\boldsymbol{x}_o$, for which we want to obtain the posterior distribution. We introduce two approaches to optimize the context.

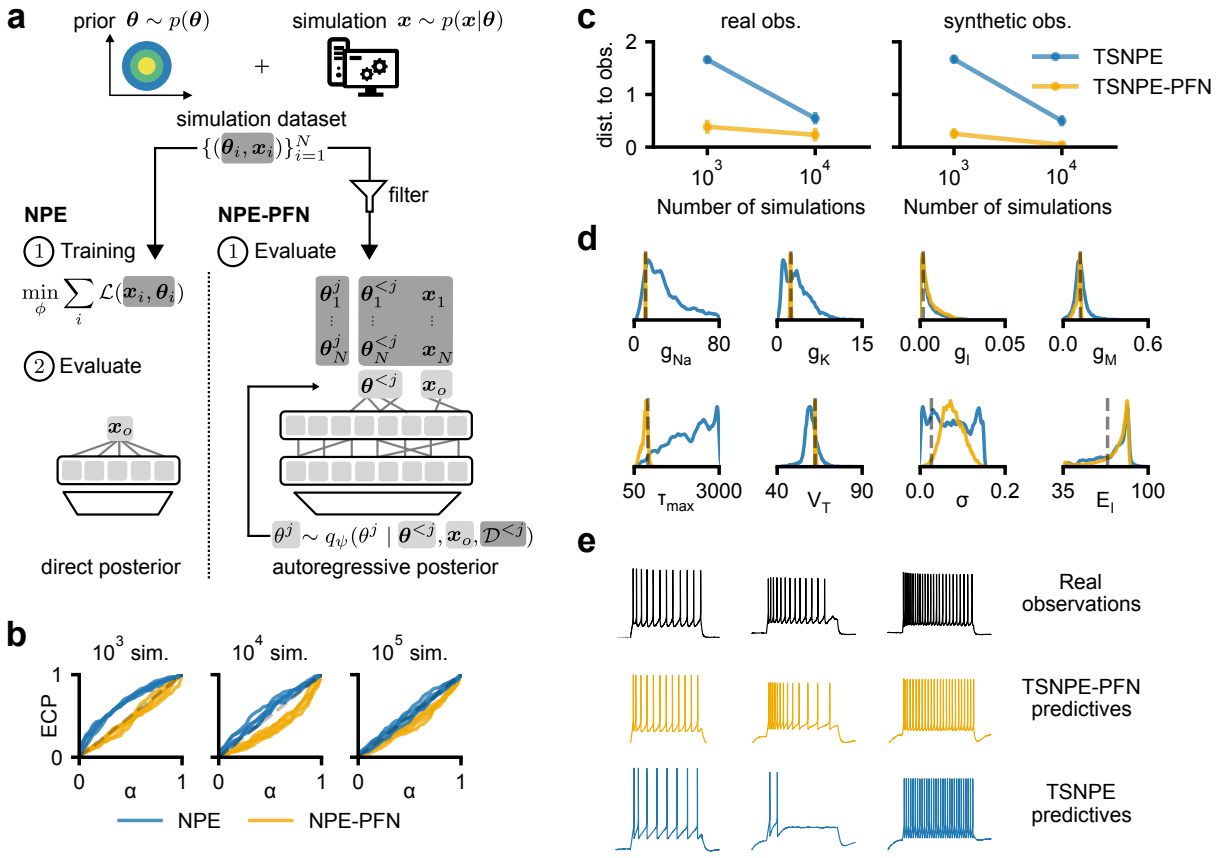


Figure 3.4: Training-free, simulation efficient Bayesian inference with TabPFN. (a) Unlike classical neural posterior estimation (NPE), NPE-PFN does not require task-specific training. Training-free inference is achieved by using the tabular foundation model TabPFN to autoregressively estimate the posterior. The simulations used to estimate the posterior can be filtered to include only those close to the given observation. This filtering step optimizes the dataset used in the limited context of TabPFN and improves performance. We apply NPE-PFN to a parameter inference task in neuroscience, where parameters of the Hodkgin-Huxley model are estimated from recordings of action potentials. **(b)** Simulation-based calibration of both NPE-PFN and flow-based NPE across different simulation budgets. Perfect calibration is given when calibration curves are on the diagonal. **(c)** Average Euclidean distance of posterior predictives to observations. Posteriors were estimated using the sequential variant of NPE-PFN, TSNPE-PFN, and again compared to the flow-based baseline. **(d)** TSNPE and TSNPE-PFN posterior marginals of an example synthetic observation. The dashed gray line indicates the ground truth parameter. **(e)** Example observations and corresponding posterior predictives from TSNPE and TSNPE-PFN posteriors. Panels adapted from Vetter et al. [162].

The first approach is filtering, where the simulations closest to a given observation are included in the context. Importantly, we show that this process does not bias the posterior estimate. Second, we use NPE-PFN with sequential inference to acquire relevant simulations for a given observation via active learning. In particular, we demonstrate that, with some modifications, TabPFN can serve as an efficient density estimator for the previously published sequential inference method TSNPE [95], resulting in TSNPE-PFN.

3.5.3 Results

We evaluate NPE-PFN and TSNPE-PFN on a variety of classical SBI benchmark tasks for amortized and sequential inference [163]. Results on these benchmarks show that (TS)NPE-PFN is especially effective with small simulation counts of 100 or 1000, while remaining competitive with baseline methods for larger simulation counts. Therefore, NPE-PFN and its sequential variant, TSNPE-PFN, are substantially more simulation efficient than previous methods and benefit from the principled approach of TabPFN for handling the small data regime. This simulation efficiency becomes especially evident when considering two real-world parameter inference tasks where the observations are neurophysiological time series.

For the first task, we perform parameter inference in a single-compartment Hodgkin-Huxley model based on action potential recordings from the Allen Cell Types Database [164]. In the second application, we consider higher-dimensional parameter spaces and estimate parameters of the pyloric network of the crab Cancer borealis [165, 166]. In both cases, we use domain-specific summary statistics to extract features from these neurophysiological time series [25, 27]. For the single compartment Hodgkin-Huxley model, NPE-PFN shows good (simulation-based) calibration [167], especially for a small simulation budget (Fig. 3.4b). Furthermore, compared to sequential SBI baseline methods, TSNPE-PFN requires orders of magnitude fewer simulations to achieve the same quality of inference, as measured by the distance between observation and posterior predictive simulations (Fig. 3.4c). In addition, the estimated posterior marginals are substantially more constrained around the ground truth when using TSNPE-PFN on synthetic observations (Fig. 3.4d). Visually, the posterior predictives of TSNPE-PFN closely match the real observations (Fig. 3.4e). Overall, our results demonstrate that (TS)NPE-PFN is an easy-to-use, highly simulation-efficient method for simulation-based inference.

3.5.4 Contributions

This publication was coauthored by me, Manuel Gloeckler, Daniel Gedon, and Jakob H. Macke. I implemented a prototype of (TS)NPE-PFN to check its feasibility. Manuel Gloeckler then set up the experimental infrastructure. Both Manuel Gloeckler and I performed the proper implementation of (TS)NPE-PFN and the associated evaluation pipeline. Throughout this process, Daniel Gedon provided feedback on the code, and also contributed code directly (including the misspecification experiment). I performed the “Allen experiment”. Manuel Gloeckler performed the “Pyloric experiment”. Manuel Gloeckler, Daniel Gedon, and I wrote the manuscript, with feedback from Jakob H. Macke. Throughout the whole project, Daniel Gedon and Jakob H. Macke provided supervision.

Chapter 4

Conclusion

This thesis aims to advance the state of the art in deep probabilistic models for physiological time series applications. To this end, I have made four contributions: We proposed models tailored to the interpretable prediction of neonatal apneas and hypopneas (Sec. 3.2), as well as data-efficient and high-fidelity generation of neurophysiological time series (Sec. 3.3). Then, we presented methods for estimating high-entropy source distributions (Sec. 3.4) and for simulation-efficient Bayesian inference (Sec. 3.5). We demonstrated the utility of these methods by applying them to challenging parameter inference problems from neuroscience. Looking back on these contributions, two overarching approaches to developing these models and methods stand out. The first three contributions rely heavily on domain knowledge, whereas the fourth contribution takes a different approach by using a general-purpose foundation model that has been pre-trained on vast amounts of data. This thesis thus reflects the broader trend in recent years of training and using increasingly large, general-purpose foundation models. To conclude this thesis, I discuss the common themes of data efficiency, interpretability and accessibility in light of these two overarching approaches.

Data efficiency

Data efficiency refers to the ability of models and methods to perform well with limited training data. As discussed, for applications involving physiological time series, data efficiency is crucial, as in many cases, large, high-quality datasets remain scarce. In particular, datasets of clinical time series are often costly to collect or subject to strict privacy restrictions. Additionally, in parameter inference, simulators of physiological processes can be computationally expensive, prohibiting the creation of large datasets of simulations to train neural inference methods.

One approach to making deep probabilistic models more data efficient is incorporating domain knowledge into their construction. In our work on denoising diffusion models for neurophysiological time series (Sec. 3.3), we incorporated neuroscientific knowledge about the multiscale nature of neural dynamics and power spectra into the diffusion backbone and diffusion process. These modifications enabled us to train high-fidelity diffusion models with only a few hundred samples, accurately capturing neuroscientific statistics of interest and outperforming previous generative models. However, incorporating domain knowledge directly into the model's construction limits its applicability to other types of time series. In our case, the proposed modifications may be less beneficial for time series with notably different temporal correlations and dependencies.

In recent years, general-purpose foundation models have emerged as an alternative approach for increasing data efficiency [168, 169]. Due to the general representations obtained during pretraining on vast amounts of diverse and non-task-specific data, foundation models often enable

highly data-efficient approaches. In this thesis, we demonstrated this concept for simulation-based inference (SBI). Our method NPE-PFN (Sec. 3.5) uses the tabular foundation model TabPFN [80] for SBI and is highly data-efficient in comparison to previous SBI methods. For parameter inference in the Hodgkin-Huxley model, the sequential variant of NPE-PFN required an order of magnitude fewer simulations to achieve comparable inference quality. Notably, as NPE-PFN uses TabPFN, it is currently limited to lower-dimensional data, and can only process high-dimensional time series if informative summary statistics of them are available.

Meanwhile, the number of time-series foundation models capable of handling increasingly high-dimensional time series is growing rapidly [170–173]. However, these models cannot be used for Bayesian inference and often focus on common time-series applications, such as forecasting and imputation. Scaling these models up to the point where a method like NPE-PFN can operate directly on high-dimensional time series presents exciting opportunities for future work. However, it is unlikely that these general-purpose models will be able to fully replace tailored approaches, particularly for complex time series applications. Therefore, future work should also focus on the development of approaches that utilize both large-scale compute and available (physiological) domain knowledge.

Interpretability

Model interpretability broadly refers to the extent to which a model's predictions can be understood. As discussed, in clinical use, interpretability is often deemed critical because it enables practitioners to trust or distrust a prediction and allows to identify spurious or biased correlations [31, 32]. It can also be useful for research purposes. For instance, when a mechanistic simulator is unavailable, interpretable models can provide insight into how the parameters of underlying physiological processes interact. In this thesis, we proposed an interpretable deep probabilistic model for apnea and hypopnea prediction (Sec. 3.2). Specifically, we used a neural additive model (NAM, 141) and tailored it to polysomnography recordings in close consultation with our clinical collaborators. The model's additive structure enabled us to determine which signal modalities and time series features of the polysomnography recording are important for a given prediction.

However, this notion of interpretability is application-dependent. Since our NAM is tailored to polysomnography recordings, its explanations are primarily meaningful in the context of apnea and hypopnea prediction. Applying a similar model to other types of time series or prediction problems may be nonsensical. This example reflects a general and controversially discussed issue: The close link between interpretability and application results in a lack of a universally accepted definition of what constitutes a sufficiently interpretable model [30, 174, 175].

Unfortunately, the growing use of foundation models for physiological time series applications exacerbates the interpretability issues. Unlike NAMs and similar methods, which are typically considered to be interpretable due to their simplicity, foundation models are uninterpretable black boxes with potentially billions of parameters. Much work has gone into providing post-hoc explanations for these black box models, typically by training a second model to provide insight into their inner workings [176, 177]. However, these explanations have been criticized for being misleading and for not accurately reflecting the model's computations [30, 178, 179].

Despite these technical and conceptual challenges, interpretability remains a desirable property of deep probabilistic models, particularly in high-stakes settings like clinical practice. Future work could, for example, focus on developing new techniques to distill larger time-series foundation models into smaller and simpler models [180–183]. However, due to the application-dependent nature of interpretability, no matter the methodology, such work should be done in close consultation

with domain experts.

Accessibility

Although accessibility has only been discussed briefly in this thesis so far, it is an important property in practice. Here, accessibility refers to how easily a machine learning method or model can be applied by practitioners. In the context of deep probabilistic models for physiological time series, this property is particularly important as clinicians and life science researchers may lack extensive machine learning expertise. Factors that limit accessibility and make models and methods challenging to use include strong dependence on many hyperparameters, the need for (several stages of) optimization and sensitivity to data (pre-)processing.

Sourcerer, our method for source distribution estimation (Sec. 3.4), is an example of a relatively complex method. It depends on a regularization hyperparameter, which users must carefully tune. Furthermore, Sourcerer requires a differentiable simulator, which is often unavailable in practice. Thus, in many cases, a tractable surrogate simulator must first be trained. As such, Sourcerer is a specialized approach that requires users to have a fair amount of machine learning expertise.

In contrast, foundation models offer a promising solution to many of the aforementioned accessibility challenges via in-context learning. For our method NPE-PFN (Sec. 3.5), we repurpose the tabular foundation model TabPFN [80] for SBI. Consequently, NPE-PFN is entirely training-free, eliminating the challenging aspects of training and hyperparameter tuning that limited the accessibility of previous SBI methods. As mentioned, beyond NPE-PFN and SBI, time-series foundation models now offer training-free alternatives for many time series applications [170]. However, it is important to note that although foundation models are user-friendly in terms of methodological complexity, their often demanding hardware requirements can present a different barrier to their use [168].

Even disregarding hardware requirements, foundation models are unlikely to fully replace tailored approaches, especially for complex problems such as source distribution estimation. Therefore, developing frameworks for automated machine learning (AutoML) [184–186] that automate challenging aspects of deep probabilistic models remain important in the future.

Together, our contributions demonstrate that tailored, task-specific approaches and approaches based on foundation models can improve the data efficiency, interpretability, and accessibility of deep probabilistic models for physiological time series applications. I hope that the contributions presented in this thesis will stimulate further research in both directions to address challenges concerning physiological time series and beyond.

References

[1] David M Mirvis and Ary L Goldberger. Electrocardiography. *Heart disease*, 1:82–128, 2001.

[2] Ernst Niedermeyer and FH Lopes da Silva. *Electroencephalography: basic principles, clinical applications, and related fields*. Lippincott Williams & Wilkins, 2005.

[3] György Buzsáki, Costas A Anastassiou, and Christof Koch. The origin of extracellular fields and currents—EEG, ECoG, LFP and spikes. *Nature reviews neuroscience*, 13(6):407–420, 2012.

[4] Ali S Afshar, Yijun Li, Zixu Chen, Yuxuan Chen, Jae Hun Lee, Darius Irani, Aidan Crank, Digvijay Singh, Michael Kanter, Nauder Faraday, et al. An exploratory data quality analysis of time series physiologic signals using a large-scale intensive care unit database. *JAMIA open*, 4(3):ooab057, 2021.

[5] Ary L Goldberger, Luis AN Amaral, Leon Glass, Jeffrey M Hausdorff, Plamen Ch Ivanov, Roger G Mark, Joseph E Mietus, George B Moody, Chung-Kang Peng, and H Eugene Stanley. Physiobank, Physiotoolkit, and Physionet: components of a new research resource for complex physiologic signals. *circulation*, 101(23):e215–e220, 2000.

[6] Daniel Jarrett, Jinsung Yoon, Ioana Bica, Zhaozhi Qian, Ari Ercole, and Mihaela van der Schaar. Clairvoyance: A pipeline toolkit for medical time series. In *International Conference on Learning Representations*, 2021.

[7] Kevin P Murphy. *Probabilistic machine learning: Advanced topics*. MIT press, 2023.

[8] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553): 436–444, 2015.

[9] Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

[10] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9 (8):1735–1780, 1997.

[11] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

[12] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[13] Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*, 2021.

[14] Jimmy TH Smith, Andrew Warrington, and Scott W Linderman. Simplified state space layers for sequence modeling. In *International Conference on Learning Representations*, 2023.

[15] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. *Data mining and knowledge discovery*, 33(4):917–963, 2019.

[16] Yannick Roy, Hubert Banville, Isabela Albuquerque, Alexandre Gramfort, Tiago H Falk, and Jocelyn Faubert. Deep learning-based electroencephalography analysis: a systematic review. *Journal of neural engineering*, 16(5):051001, 2019.

[17] Pranav Rajpurkar, Awni Y Hannun, Masoumeh Haghpanahi, Codie Bourn, and Andrew Y Ng. Cardiologist-level arrhythmia detection with convolutional neural networks. *arXiv preprint arXiv:1707.01836*, 2017.

[18] Vernon J Lawhern, Amelia J Solon, Nicholas R Waytowich, Stephen M Gordon, Chou P Hung, and Brent J Lance. EEGnet: a compact convolutional neural network for EEG-based brain–computer interfaces. *Journal of neural engineering*, 15(5):056013, 2018.

[19] Yusuke Tashiro, Jiaming Song, Yang Song, and Stefano Ermon. CSDI: Conditional score-based diffusion models for probabilistic time series imputation. *Advances in neural information processing systems*, 34:24804–24816, 2021.

[20] Juan Miguel Lopez Alcaraz and Nils Strodthoff. Diffusion-based time series imputation and forecasting with structured state space models. *Transactions on machine learning research*, pages 1–36, 2023.

[21] Jinsung Yoon, James Jordon, and Mihaela Schaar. GAIN: Missing data imputation using generative adversarial nets. In *International conference on machine learning*, pages 5689–5698, 2018.

[22] Vincent Fortuin, Dmitry Baranchuk, Gunnar Rätsch, and Stephan Mandt. GP-VAE: Deep probabilistic time series imputation. In *International Conference on Artificial Intelligence and Statistics*, pages 1651–1661, 2020.

[23] Juan Miguel Lopez Alcaraz and Nils Strodthoff. Diffusion-based conditional ECG generation with structured state space models. *Computers in biology and medicine*, 163:107115, 2023.

[24] Chethan Pandarinath, Daniel J O'Shea, Jasmine Collins, Rafal Jozefowicz, Sergey D Stavisky, Jonathan C Kao, Eric M Trautmann, Matthew T Kaufman, Stephen I Ryu, Leigh R Hochberg, et al. Inferring single-trial neural population dynamics using sequential auto-encoders. *Nature methods*, 15(10):805–815, 2018.

[25] 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.

[26] Ding Zhou and Xue-Xin Wei. Learning identifiable and interpretable latent models of high-dimensional neural activity using pi-VAE. *Advances in Neural Information Processing Systems*, 33:7234–7247, 2020.

[27] Michael Deistler, Jakob H Macke, and Pedro J Gonçalves. Energy-efficient network activity from disparate circuit parameters. *Proceedings of the National Academy of Sciences*, 119(44): e2207632119, 2022.

[28] Rabia Gondur, Usama Bin Sikandar, Evan Schaffer, Mikio Christian Aoi, and Stephen L Keeley. Multi-modal Gaussian process variational autoencoders for neural and behavioral data. In *International Conference on Learning Representations*, 2024.

[29] Michael I Jordan and Tom M Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, 2015.

[30] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature machine intelligence*, 1(5):206–215, 2019.

[31] Christoph Molnar, Giuseppe Casalicchio, and Bernd Bischl. Interpretable machine learning–a brief history, state-of-the-art and challenges. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 417–431, 2020.

[32] Susu Sun, Lisa M Koch, and Christian F Baumgartner. Right for the wrong reason: Can interpretable ML techniques detect spurious correlations? In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 425–434, 2023.

[33] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[34] Elisabeth Bloch-Salisbury, Premananda Indic, Frank Bednarek, and David Paydarfar. Stabilizing immature breathing patterns of preterm infants using stochastic mechanosensory stimulation. *Journal of Applied Physiology*, 107(4):1017–1027, 2009.

[35] Miguel Altuve, Guy Carrault, Alain Beuchee, Patrick Pladys, and Alfredo I Hernandez. Online apnea-bradycardia detection using hidden semi-Markov models. In *Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 4374–4377, 2011.

[36] Brooke D Vergales, Alix O Paget-Brown, Hoshik Lee, Lauren E Guin, Terri J Smoot, Craig G Rusin, Matthew T Clark, John B Delos, Karen D Fairchild, Douglas E Lake, et al. Accurate automated apnea analysis in preterm infants. *American journal of perinatology*, 31(02): 157–162, 2014.

[37] Luca Cattani, Davide Alinovi, Gianluigi Ferrari, Riccardo Raheli, Elena Pavlidis, Carlotta Spagnoli, and Francesco Pisani. Monitoring infants by automatic video processing: A unified approach to motion analysis. *Computers in biology and Medicine*, 80:158–165, 2017.

[38] Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. Diffusion models: A comprehensive survey of methods and applications. *ACM computing surveys*, 56(4):1–39, 2023.

[39] Tijin Yan, Hongwei Zhang, Tong Zhou, Yufeng Zhan, and Yuanqing Xia. ScoreGrad: Multivariate probabilistic time series forecasting with continuous energy-based generative models. *arXiv preprint arXiv:2106.10121*, 2021.

[40] Kashif Rasul, Calvin Seward, Ingmar Schuster, and Roland Vollgraf. Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting. In *International conference on machine learning*, pages 8857–8868, 2021.

[41] 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.

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

[43] George Papamakarios, David Sterratt, and Iain Murray. Sequential neural likelihood: Fast likelihood-free inference with autoregressive flows. In *International Conference on Artificial Intelligence and Statistics*, pages 837–848, 2019.

[44] Manuel Gloeckler, Michael Deistler, Christian Dietrich Weilbach, Frank Wood, and Jakob H. Macke. All-in-one simulation-based inference. In *International Conference on Machine Learning*, volume 235, pages 15735–15766, 2024.

[45] Maxime Vandegar, Michael Kagan, Antoine Wehenkel, and Gilles Louppe. Neural empirical Bayes: Source distribution estimation and its applications to simulation-based inference. In *International Conference on Artificial Intelligence and Statistics*, pages 2107–2115, 2021.

[46] François Rozet, Gérôme Andry, François Lanusse, and Gilles Louppe. Learning diffusion priors from observations by expectation maximization. *Advances in Neural Information Processing Systems*, 37:87647–87682, 2024.

[47] Jonas Arruda, Yannik Schälte, Clemens Peiter, Olga Teplytska, Ulrich Jaehde, and Jan Hasenauer. An amortized approach to non-linear mixed-effects modeling based on neural posterior estimation. In *International Conference on Machine Learning*, 2024.

[48] Gintare Karolina Dziugaite, Daniel M Roy, and Zoubin Ghahramani. Training generative neural networks via maximum mean discrepancy optimization. In *Uncertainty in Artificial Intelligence*, 2015.

[49] Kimia Nadjahi, Alain Durmus, Lénaïc Chizat, Soheil Kolouri, Shahin Shahrampour, and Umut Simsekli. Statistical and topological properties of sliced probability divergences. *Advances in Neural Information Processing Systems*, 33:20802–20812, 2020.

[50] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[51] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25, 2012.

[52] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[53] Zachi I Attia, Suraj Kapa, Francisco Lopez-Jimenez, Paul M McKie, Dorothy J Ladewig, Gaurav Satam, Patricia A Pellikka, Maurice Enriquez-Sarano, Peter A Noseworthy, Thomas M Munger, et al. Screening for cardiac contractile dysfunction using an artificial intelligence–enabled electrocardiogram. *Nature medicine*, 25(1):70–74, 2019.

[54] AH Ribeiro, MH Ribeiro, GM Paixão, DM Oliveira, PR Gomes, JA Canazart, MP Ferreira, CR Andersson, PW Macfarlane, W Meira Jr, et al. Automatic diagnosis of the 12-lead ECG using a deep neural network. *Nature communications*, 11:1–9, 2020.

[55] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[56] Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*, 2014.

[57] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. *International Conference on Machine Learning*, pages 1530–1538, 2015.

[58] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. *Advances in neural information processing systems*, 31, 2018.

[59] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using RealNVP. In *International Conference on Learning Representations*, 2017.

[60] George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. *Advances in neural information processing systems*, 30, 2017.

[61] Shuangfei Zhai, Ruixiang Zhang, Preetum Nakkiran, David Berthelot, Jiatao Gu, Huangjie Zheng, Tianrong Chen, Miguel Angel Bautista, Navdeep Jaitly, and Josh Susskind. Normalizing flows are capable generative models. In *Forty-second International Conference on Machine Learning*, 2025.

[62] Andrew Ng and Michael Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *Advances in Neural Information Processing Systems*, 14, 2001.

[63] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265, 2015.

[64] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[65] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021.

[66] Yang Song and Prafulla Dhariwal. Improved techniques for training consistency models. In *International Conference on Learning Representations*, 2024.

[67] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *International Conference on Machine Learning*, 2023.

[68] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. DPM-solver: A fast ODE solver for diffusion probabilistic model sampling in around 10 steps. *Advances in Neural Information Processing Systems*, 35:5775–5787, 2022.

[69] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021.

[70] Ruiqi Gao, Emiel Hoogeboom, Jonathan Heek, Valentin De Bortoli, Kevin Patrick Murphy, and Tim Salimans. Diffusion models and Gaussian flow matching: Two sides of the same coin. In *The Fourth Blogpost Track at ICLR 2025*, 2025.

[71] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[72] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in Neural Information Processing Systems*, 35:36479–36494, 2022.

[73] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[74] Samuel Müller, Noah Hollmann, Sebastian Pineda Arango, Josif Grabocka, and Frank Hutter. Transformers can do Bayesian inference. In *International Conference on Learning Representations*, 2022.

[75] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[76] Thomas Nagler. Statistical foundations of prior-data fitted networks. In *International Conference on Machine Learning*, pages 25660–25676, 2023.

[77] Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. TabPFN: A transformer that solves small tabular classification problems in a second. In *NeurIPS 2022 First Table Representation Workshop*, 2022.

[78] Jingang Qu, David Holzmüller, Gael Varoquaux, and Marine Le Morvan. TabICL: A tabular foundation model for in-context learning on large data. In *International Conference on Machine Learning*, 2025.

[79] Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. *Elements of causal inference: foundations and learning algorithms*. The MIT press, 2017.

[80] Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister, and Frank Hutter. Accurate predictions on small data with a tabular foundation model. *Nature*, 637(8045):319–326, 2025.

[81] Benjamin P Abbott, Richard Abbott, Thomas D Abbott, Matthew R Abernathy, Fausto Acernese, Kendall Ackley, Carl Adams, Thomas Adams, Paolo Addesso, Rana X Adhikari, et al. Observation of gravitational waves from a binary black hole merger. *Physical review letters*, 116(6):061102, 2016.

[82] Mark A Beaumont, Wenyang Zhang, and David J Balding. Approximate Bayesian computation in population genetics. *Genetics*, 162(4):2025–2035, 2002.

[83] Maximilian Dax, Stephen R Green, Jonathan Gair, Nihar Gupte, Michael Pürrer, Vivien Raymond, Jonas Wildberger, Jakob H Macke, Alessandra Buonanno, and Bernhard Schölkopf. Real-time inference for binary neutron star mergers using machine learning. *Nature*, 639 (8053):49–53, 2025.

[84] Johann Brehmer and Kyle Cranmer. Simulation-based inference methods for particle physics. *Artificial Intelligence for High Energy Physics*, pages 579–611, 2022.

[85] Andrew Gelman, John B Carlin, Hal S Stern, and Donald B Rubin. *Bayesian data analysis*. Chapman and Hall/CRC, 1995.

[86] Matthew D Hoffman, Andrew Gelman, et al. The No-U-Turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, 15(1): 1593–1623, 2014.

[87] Arnaud Doucet, Nando De Freitas, and Neil Gordon. An introduction to sequential Monte Carlo methods. *Sequential Monte Carlo methods in practice*, pages 3–14, 2001.

[88] Tomas Geffner, George Papamakarios, and Andriy Mnih. Compositional score modeling for simulation-based inference. In *International Conference on Machine Learning*, pages 11098–11116, 2023.

[89] Louis Sharrock, Jack Simons, Song Liu, and Mark Beaumont. Sequential neural score estimation: Likelihood-free inference with conditional score based diffusion models. In *International Conference on Machine Learning*, pages 44565–44602, 2024.

[90] Conor Durkan, Iain Murray, and George Papamakarios. On contrastive learning for likelihood-free inference. In *International Conference on Machine Learning*, pages 2771–2781, 2020.

[91] Benjamin K Miller, Christoph Weniger, and Patrick Forré. Contrastive neural ratio estimatizon. In *Advances in Neural Information Processing Systems*, volume 35, pages 3262–3278, 2022.

[92] Paul E Chang, Nasrulloh Loka, Daolang Huang, Ulpu Remes, Samuel Kaski, and Luigi Acerbi. Amortized probabilistic conditioning for optimization, simulation and inference. In *International Conference on Artificial Intelligence and Statistics*, 2025.

[93] Scott A Sisson, Yanan Fan, and Mark Beaumont. *Handbook of approximate Bayesian computation*. CRC press, 2018.

[94] Joeri Hermans, Arnaud Delaunoy, François Rozet, Antoine Wehenkel, and Gilles Louppe. A crisis in simulation-based inference? Beware, your posterior approximations can be unfaithful. *Transactions on Machine Learning Research*, 2022.

[95] 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.

[96] Manuel Glöckler, Michael Deistler, and Jakob H Macke. Variational methods for simulation-based inference. In *International Conference on Learning Representations*, 2022.

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

[98] T Butler, J Jakeman, and Tim Wildey. Combining push-forward measures and Bayes' rule to construct consistent solutions to stochastic inverse problems. *SIAM Journal on Scientific Computing*, 40(2):A984–A1011, 2018.

[99] Herbert E Robbins. An empirical Bayes approach to statistics. *Breakthroughs in Statistics: Foundations and basic theory*, pages 388–394, 1992.

[100] George Casella. An introduction to empirical Bayes data analysis. *The American Statistician*, 39(2):83–87, 1985.

[101] Petrus Mikkola, Osvaldo A Martin, Suyog Chandramouli, Marcelo Hartmann, Oriol Abril Pla, Owen Thomas, Henri Pesonen, Jukka Corander, Aki Vehtari, Samuel Kaski, et al. Prior knowledge elicitation: The past, present, and future. *Bayesian Analysis*, 19(4):1129–1161, 2024.

[102] Gilles Louppe, Joeri Hermans, and Kyle Cranmer. Adversarial variational optimization of non-differentiable simulators. In *International Conference on Artificial Intelligence and Statistics*, pages 1438–1447, 2019.

[103] Mathias Backes, Anja Butter, Monica Dunford, and Bogdan Malaescu. An unfolding method based on conditional invertible neural networks (cinn) using iterative training. *SciPost Physics Core*, 7(1):007, 2024.

[104] Anders Andreassen, Patrick T Komiske, Eric M Metodiev, Benjamin Nachman, and Jesse Thaler. OmniFold: A method to simultaneously unfold all observables. *Physical review letters*, 124(18):182001, 2020.

[105] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[106] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32, 2019.

[107] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[108] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference for Learning Representations*, 2015.

[109] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference for Learning Representations*, 2019.

[110] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

[111] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (GELUs). *arXiv preprint arXiv:1606.08415*, 2016.

[112] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[113] Hossein Gholamalinezhad and Hossein Khosravi. Pooling methods in deep neural networks, a review. *arXiv preprint arXiv:2009.07485*, 2020.

[114] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[115] Satya Narayan Shukla and Benjamin M Marlin. Multi-time attention networks for irregularly sampled time series. In *International Conference on Learning Representations*, 2021.

[116] Jiawen Zhang, Shun Zheng, Wei Cao, Jiang Bian, and Jia Li. Warpformer: A multi-scale modeling approach for irregular clinical time series. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 3273–3285, 2023.

[117] Hava T Siegelmann and Eduardo D Sontag. Turing computability with neural nets. *Applied Mathematics Letters*, 4(6):77–80, 1991.

[118] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pages 1310–1318, 2013.

[119] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing*, 2014.

[120] Zhengping Che, Sanjay Purushotham, Kyunghyun Cho, David Sontag, and Yan Liu. Recurrent neural networks for multivariate time series with missing values. *Scientific reports*, 8(1): 6085, 2018.

[121] Jinsung Yoon, Daniel Jarrett, and Mihaela Van der Schaar. Time-series generative adversarial networks. *Advances in Neural Information Processing Systems*, 32, 2019.

[122] Michael Hüsken and Peter Stagge. Recurrent neural networks for time series classification. *Neurocomputing*, 50:223–235, 2003.

[123] Zhiguang Wang, Weizhong Yan, and Tim Oates. Time series classification from scratch with deep neural networks: A strong baseline. In *2017 International joint conference on neural networks (IJCNN)*, pages 1578–1585, 2017.

[124] Bendong Zhao, Huanzhang Lu, Shangfeng Chen, Junliang Liu, and Dongya Wu. Convolutional neural networks for time series classification. *Journal of systems engineering and electronics*, 28(1):162–169, 2017.

[125] Alessio Petrozziello, Christopher WG Redman, Aris T Papageorghiou, Ivan Jordanov, and Antoniya Georgieva. Multimodal convolutional neural networks to detect fetal compromise during labor and delivery. *IEEE Access*, 7:112026–112036, 2019.

[126] Aaron Van Den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, Koray Kavukcuoglu, et al. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 12, 2016.

[127] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

[128] Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. Long range arena: A benchmark for efficient transformers. In *International Conference on Learning Representations*, 2021.

[129] Albert Gu, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Ré. Hippo: Recurrent memory with optimal polynomial projections. *Advances in neural information processing systems*, 33:1474–1487, 2020.

[130] Karan Goel, Albert Gu, Chris Donahue, and Christopher Ré. It's raw! Audio generation with state-space models. In *International conference on machine learning*, pages 7616–7633, 2022.

[131] Daniel Y Fu, Tri Dao, Khaled K Saab, Armin W Thomas, Atri Rudra, and Christopher Ré. Hungry hungry hippos: Towards language modeling with state space models. In *International Conference on Learning Representations*, 2023.

[132] Daniel Y Fu, Elliot L Epstein, Eric Nguyen, Armin W Thomas, Michael Zhang, Tri Dao, Atri Rudra, and Christopher Ré. Simple hardware-efficient long convolutions for sequence modeling. In *International Conference on Machine Learning*, pages 10373–10391, 2023.

[133] Yuhong Li, Tianle Cai, Yi Zhang, Deming Chen, and Debadeepta Dey. What makes convolutional models great on long sequence modeling? In *International Conference on Learning Representations*, 2023.

[134] Trent Henderson and Ben D Fulcher. An empirical evaluation of time-series feature sets. In *2021 International Conference on Data Mining Workshops (ICDMW)*, pages 1032–1038, 2021.

[135] Martin Pospischil, Maria Toledo-Rodriguez, Cyril Monier, Zuzanna Piwkowska, Thierry Bal, Yves Frégnac, Henry Markram, and Alain Destexhe. Minimal Hodgkin–Huxley type models for different classes of cortical and thalamic neurons. *Biological cybernetics*, 99(4):427–441, 2008.

[136] Patricia Pauli, Anne Koch, Julian Berberich, Paul Kohler, and Frank Allgöwer. Training robust neural networks using Lipschitz bounds. *IEEE Control Systems Letters*, 6:121–126, 2021.

[137] Lukas Schott, Jonas Rauber, Matthias Bethge, and Wieland Brendel. Towards the first adversarially robust neural network model on MNIST. In *International Conference on Learning Representations*, 2019.

[138] Ian Zuzarte, Dagmar Sternad, and David Paydarfar. Predicting apneic events in preterm infants using cardio-respiratory and movement features. *Computer methods and programs in biomedicine*, 209:106321, 2021.

[139] G Pravisani, A Beuchee, Luca Mainardi, and G Carrault. Short term prediction of severe bradycardia in premature newborns. In *Computers in Cardiology*, pages 725–728, 2003.

[140] Julius Vetter, Kathleen Lim, Tjeerd MH Dijkstra, Peter A Dargaville, Oliver Kohlbacher, Jakob H Macke, and Christian F Poets. Neonatal apnea and hypopnea prediction in infants with Rfobin sequence with neural additive models for time series. *PLOS Digital Health*, 3(12):e0000678, 2024.

[141] Rishabh Agarwal, Levi Melnick, Nicholas Frosst, Xuezhou Zhang, Ben Lengerich, Rich Caruana, and Geoffrey E Hinton. Neural additive models: Interpretable machine learning with neural nets. *Advances in Neural Information Processing Systems*, 34:4699–4711, 2021.

[142] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2921–2929, 2016.

[143] Corstiaan C Breugem, Kelly N Evans, Christian F Poets, Sunjay Suri, Arnaud Picard, Charles Filip, Emma C Paes, Felicity V Mehendale, Howard M Saal, Hanneke Basart, et al. Best practices for the diagnosis and evaluation of infants with Robin sequence: A clinical consensus report. *JAMA pediatrics*, 170(9):894–902, 2016.

[144] Kathleen Lim, Mirja Quante, Tjeerd MH Dijkstra, Gabriele Hilbert-Moessner, Cornelia Wiechers, Peter Dargaville, and Christian F Poets. Should obstructive hypopneas be included when analyzing sleep studies in infants with Robin sequence? *Sleep Medicine*, 98:9–12, 2022.

[145] Tom Fawcett. An introduction to ROC analysis. *Pattern recognition letters*, 27(8):861–874, 2006.

[146] Caspar Meijer and Lydia Y Chen. The rise of diffusion models in time-series forecasting. *arXiv preprint arXiv:2401.03006*, 2024.

[147] Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. DiffWave: A versatile diffusion model for audio synthesis. In *International Conference on Learning Representations*, 2021.

[148] Julius Vetter, Jakob H Macke, and Richard Gao. Generating realistic neurophysiological time series with denoising diffusion probabilistic models. *Patterns*, 5(9), 2024.

[149] Carmen Varela and Matthew A Wilson. mPFC spindle cycles organize sparse thalamic activation and recently active CA1 cells during non-REM sleep. *eLife*, 9:e48881, 2020.

[150] Victor Venema, Felix Ament, and Clemens Simmer. A stochastic iterative amplitude adjusted Fourier transform algorithm with improved accuracy. *Nonlinear Processes in Geophysics*, 13 (3):321–328, 2006.

[151] Steven M Peterson, Satpreet H Singh, Benjamin Dichter, Michael Scheid, Rajesh PN Rao, and Bingni W Brunton. AJILE12: Long-term naturalistic human intracranial neural recordings and pose. *Scientific data*, 9(1):184, 2022.

[152] Toru Yanagawa, Zenas C Chao, Naomi Hasegawa, and Naotaka Fujii. Large-scale information flow in conscious and unconscious states: An ECoG study in monkeys. *PloS one*, 8(11): e80845, 2013.

[153] Jaivardhan Kapoor, Auguste Schulz, Julius Vetter, Felix Pei, Richard Gao, and Jakob H Macke. Latent diffusion for neural spiking data. *Advances in Neural Information Processing Systems*, 37:118119–118154, 2024.

[154] Irving J Good. Maximum entropy for hypothesis formulation, especially for multidimensional contingency tables. *The Annals of Mathematical Statistics*, 34(3):911–934, 1963.

[155] Edwin T Jaynes. Prior probabilities. *IEEE Transactions on systems science and cybernetics*, 4 (3):227–241, 2007.

[156] Julius Vetter, Guy Moss, Cornelius Schröder, Richard Gao, and Jakob H Macke. Sourcerer: Sample-based maximum entropy source distribution estimation. *Advances in Neural Information Processing Systems*, 37, 2024.

[157] Kimia Nadjahi. *Sliced-Wasserstein distance for large-scale machine learning: theory, methodology and extensions*. PhD thesis, Institut polytechnique de Paris, 2021.

[158] Thomas B Berrett, Richard J Samworth, and Ming Yuan. Efficient multivariate entropy estimation via k-nearest neighbour distances. *Annals of Statistics*, 2019.

[159] David Lopez-Paz and Maxime Oquab. Revisiting classifier two-sample tests. In *International Conference on Learning Representations*, 2017.

[160] Manuel Gloeckler, Michael Deistler, and Jakob H Macke. Adversarial robustness of amortized Bayesian inference. In *International Conference on Machine Learning*, pages 11493–11524, 2023.

[161] Yves Bernaerts, Michael Deistler, Pedro J Goncalves, Jonas Beck, Marcel Stimberg, Federico Scala, Andreas S Tolias, Jakob H Macke, Dmitry Kobak, and Philipp Berens. Combined statistical-biophysical modeling links ion channel genes to physiology of cortical neuron types. *Patterns*, 2025.

[162] Julius Vetter, Manuel Gloeckler, Daniel Gedon, and Jakob H Macke. Effortless, simulation-efficient bayesian inference using tabular foundation models. *Advances in Neural Information Processing Systems*, 38, 2025.

[163] Jan-Matthis Lueckmann, Jan Boelts, David Greenberg, Pedro Goncalves, and Jakob Macke. Benchmarking simulation-based inference. In *International Conference on Artificial Intelligence and Statistics*, pages 343–351, 2021.

[164] Allen Institute for Brain Science. Allen cell types database, 2015.

[165] Astrid A Prinz, Dirk Bucher, and Eve Marder. Similar network activity from disparate circuit parameters. *Nature neuroscience*, 7(12):1345–1352, 2004.

[166] Astrid A Prinz, Cyrus P Billimoria, and Eve Marder. Alternative to hand-tuning conductance-based models: construction and analysis of databases of model neurons. *Journal of neurophysiology*, 2003.

[167] Sean Talts, Michael Betancourt, Daniel Simpson, Aki Vehtari, and Andrew Gelman. Validating Bayesian inference algorithms with simulation-based calibration. *arXiv preprint arXiv:1804.06788*, 2018.

[168] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. In *International Conference on Neural Information Processing Systems*, pages 30016–30030, 2022.

[169] Ce Zhou, Qian Li, Chen Li, Jun Yu, Yixin Liu, Guangjing Wang, Kai Zhang, Cheng Ji, Qiben Yan, Lifang He, et al. A comprehensive survey on pretrained foundation models: A history from BERT to ChatGPT. *International Journal of Machine Learning and Cybernetics*, pages 1–65, 2024.

[170] Yuxuan Liang, Haomin Wen, Yuqi Nie, Yushan Jiang, Ming Jin, Dongjin Song, Shirui Pan, and Qingsong Wen. Foundation models for time series analysis: A tutorial and survey. In *Proceedings of the 30th ACM SIGKDD conference on knowledge discovery and data mining*, pages 6555–6565, 2024.

[171] Xiaoming Shi, Shiyu Wang, Yuqi Nie, Dianqi Li, Zhou Ye, Qingsong Wen, and Ming Jin. Time-MoE: Billion-scale time series foundation models with mixture of experts. In *International Conference on Learning Representations*, 2025.

[172] Azul Garza, Cristian Challu, and Max Mergenthaler-Canseco. TimeGPT-1. *arXiv preprint arXiv:2310.03589*, 2023.

[173] Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. In *Forty-first International Conference on Machine Learning*, 2024.

[174] Zachary C Lipton. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*, 16(3):31–57, 2018.

[175] Been Kim, Rajiv Khanna, and Oluwasanmi O Koyejo. Examples are not enough, learn to criticize! Criticism for interpretability. *Advances in Neural Information Processing Systems*, 29, 2016.

[176] Dang Minh, H Xiang Wang, Y Fen Li, and Tan N Nguyen. Explainable artificial intelligence: a comprehensive review. *Artificial Intelligence Review*, 55(5):3503–3568, 2022.

[177] Erico Tjoa and Cuntai Guan. A survey on explainable artificial intelligence (XAI): Toward medical XAI. *IEEE transactions on neural networks and learning systems*, 32(11):4793–4813, 2020.

[178] Lesia Semenova, Cynthia Rudin, and Ronald Parr. On the existence of simpler machine learning models. In *Proceedings of the ACM Conference on Fairness, Accountability, and Transparency*, pages 1827–1858, 2022.

[179] Sebastian Bordt, Michèle Finck, Eric Raidl, and Ulrike Von Luxburg. Post-hoc explanations fail to achieve their purpose in adversarial contexts. In *Proceedings of the ACM Conference on Fairness, Accountability, and Transparency*, pages 891–905, 2022.

[180] Li Ji-An, Marcus K Benna, and Marcelo G Mattar. Discovering cognitive strategies with tiny recurrent neural networks. *Nature*, pages 1–9, 2025.

[181] Hyeongrok Han, Siwon Kim, Hyun-Soo Choi, and Sungroh Yoon. On the impact of knowledge distillation for model interpretability. In *International Conference on Machine Learning*, 2023.

[182] Xuan Liu, Xiaoguang Wang, and Stan Matwin. Improving the interpretability of deep neural networks with knowledge distillation. In *IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 905–912, 2018.

[183] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[184] Xin He, Kaiyong Zhao, and Xiaowen Chu. AutoML: A survey of the state-of-the-art. *Knowledge-based systems*, 212:106622, 2021.

[185] Oleksandr Shchur, Ali Caner Turkmen, Nick Erickson, Huibin Shen, Alexander Shirkov, Tony Hu, and Bernie Wang. AutoGluon–TimeSeries: AutoML for probabilistic time series forecasting. In *International Conference on Automated Machine Learning*, pages 9–1, 2023.

[186] Difan Deng, Florian Karl, Frank Hutter, Bernd Bischl, and Marius Lindauer. Efficient automated deep learning for time series forecasting. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 664–680, 2022.

Appendices

PLOS DIGITAL HEALTH

RESEARCH ARTICLE

Neonatal apnea and hypopnea prediction in infants with Robin sequence with neural additive models for time series

Julius Vetter[1,2]*, **Kathleen Lim**[3,4], **Tjeerd M. H. Dijkstra**[5,6], **Peter A. Dargaville**[3,7], **Oliver Kohlbacher**[2,5,8], **Jakob H. Macke**[1,2,9], **Christian F. Poets**[4]

1 Machine Learning in Science, University of Tübingen and Tübingen AI Center, Tübingen, Germany, **2** Department of Computer Science, University of Tübingen, Tübingen, Germany, **3** Menzies Institute for Medical Research, College of Health and Medicine, University of Tasmania, Hobart, Tasmania, Australia, **4** Department of Neonatology, University Hospital Tübingen, Tübingen, Germany, **5** Institute for Translational Bioinformatics, University Hospital Tübingen, Tübingen, Germany, **6** Department of Women's Health, University Hospital Tübingen, Tübingen, Germany, **7** Neonatal and Pediatric Intensive Care Unit, Department of Pediatrics, Royal Hobart Hospital, Hobart, Tasmania, Australia, **8** Institute for Bioinformatics and Medical Informatics, University of Tübingen, Tübingen, Germany, **9** Max Planck Institute for Intelligent Systems, Tübingen, Germany

* julius.vetter@uni-tuebingen.de

Citation: Vetter J, Lim K, Dijkstra TMH, Dargaville PA, Kohlbacher O, Macke JH, et al. (2024) Neonatal apnea and hypopnea prediction in infants with Robin sequence with neural additive models for time series. PLOS Digit Health 3(12): e0000678. https://doi.org/10.1371/journal.pdig.0000678

Editor: Henry Horng-Shing Lu, National Yang Ming Chiao Tung University, TAIWAN

Received: December 11, 2023

Accepted: October 23, 2024

Published: December 13, 2024

Peer Review History: PLOS recognizes the benefits of transparency in the peer review process; therefore, we enable the publication of all of the content of peer review and author responses alongside final, published articles. The editorial history of this article is available here: https://doi.org/10.1371/journal.pdig.0000678

Data Availability Statement: The dataset analyzed in this study is available on Zenodo: https://zenodo.org/record/7711137. The code for all experiments

Abstract

Neonatal apneas and hypopneas present a serious risk for healthy infant development. Treating these adverse events requires frequent manual stimulation by skilled personnel, which can lead to alarm fatigue. This study aims to develop and validate an interpretable model that can predict apneas and hypopneas. Automatically predicting these adverse events before they occur would enable the use of methods for automatic intervention. We propose a neural additive model to predict individual occurrences of neonatal apnea and hypopnea and apply it to a physiological dataset from infants with Robin sequence at risk of upper airway obstruction. The dataset will be made publicly available together with this study. Our proposed model allows the prediction of individual apneas and hypopneas, achieving an average AuROC of 0.80 when discriminating segments of polysomnography recordings starting 15 seconds before the onset of apneas and hypopneas from control segments. Its additive nature makes the model inherently interpretable, which allowed insights into how important a given signal modality is for prediction and which patterns in the signal are discriminative. For our problem of predicting apneas and hypopneas in infants with Robin sequence, prior irregularities in breathing-related modalities as well as decreases in SpO_2 levels were especially discriminative. Our prediction model presents a step towards an automatic prediction of neonatal apneas and hypopneas in infants at risk for upper airway obstruction. Together with the publicly released dataset, it has the potential to facilitate the development and application of methods for automatic intervention in clinical practice.

is available at https://github.com/mackelab/neonatal_apnea_prediction.

Funding: This work was supported by the German Research Foundation (DFG) through Germany's Excellence Strategy, EXC number 2064/1–390727645 and SFB1233 (PN 276693517); and the German Federal Ministry of Education and Research (BMBF): Tübingen AI Center, FKZ: 01IS18039A. The funders played no role in study design, data collection, analysis and interpretation of data, or the writing of this manuscript.

Competing interests: The authors have declared that no competing interests exist.

Author summary

Neonatal apneas (pauses in breathing) and hypopneas (shallow breathing) can severely impair infant development, especially in infants with conditions such as Robin sequence that increase the risk of upper airway obstruction. These breathing problems currently require frequent manual intervention by healthcare professionals, which can lead to alarm fatigue and challenges in providing consistent care. Here, we developed an interpretable machine learning model to predict these events in advance based on polysomnographie recordings. Our model achieved an average area under the receiver operating characteristic curve (AuROC) of 0.80 when discriminating breathing 15 seconds before apnea and hypopnea events from normal breathing. In addition, the interpretable additive structure of the model provided insight into which signal modalities of the polysomnographie recording were most predictive of adverse events. As such, this study provides a step toward automated solutions in neonatal care that could potentially improve safety and reduce the burden on healthcare providers in the future.

Introduction

It would be of great clinical importance to be able to predict apnea and hypopnea events in neonates before they occur in order to perform preventive automated intervention. Automated intervention could be realized, for example, by an inflatable mattress through which stochastic vibrotactile stimulation can be applied [1].

Compared to the amount of work in automatic apnea *detection* [2–5], there has been considerably less work in automatic apnea and hypopnea *prediction* [6]. Prior work on prediction has mainly focused on associated events of bradycardia and used cardio-respiratory features together with different classifiers like hierarchical classification methods [7] and random forests [8]. Recent studies also used infant movement as an additional feature [6]. Another study more generally used deep neural networks to identify infants susceptible to apnea and hypopnea [9].

However, previous work did not address predicting individual events of apnea or hypopnea, but rather tried to predict whole episodes with repeated apneas and hypopneas [6]. Consequently, the prediction horizon for the cited approaches was on the order of several minutes [10, 11].

In this study, we were instead interested in predicting individual apnea and hypopnea events and thus worked on a time scale of seconds. Because of this new time scale, there were no readily available or traditionally used features to extract from the recorded signals. In recent years, deep neural networks have had considerable success in automating the feature extraction process. Many areas of science, including the prediction of adverse events in medical time series, have benefited from this progress [12–14]. However, a major drawback of classical deep neural network architectures is that they are a blackbox: Their complexity makes it difficult to understand why a particular prediction was made and which features of the signals contributed to it. Especially in the medical domain, where mistakes are costly, opening this blackbox and making its decisions more interpretable is crucial for applications. There are several methods to generate post-hoc explanations of black-box models, but their use in high-risk applications has recently been discouraged due to unreliable or misleading explanations [15, 16]. An alternative to post-hoc explanations are models that are interpretable by construction. Creating such models is often possible without sacrificing model performance [17].

Our contribution

We built on recent work in neural additive models [18], which form their prediction by summing over the output of multiple neural networks. The additive nature allows users to inspect the magnitude of the additive contributions for each prediction to gauge the importance of the underlying feature or signal modality. This inherent interpretability makes neural additive models a good choice for clinical prediction tasks such as individual apnea or hypopnea prediction. Since our data came in the form of time series, we replaced classically used neural network architectures with those tailored to time series. We show that our model achieved a high level of performance, with an average area under the receiver operating characteristic curve (AuROC) of 0.80 for the task of apnea and hypopnea prediction for infants with Robin sequence. In addition, our model allowed us to assess the importance of different signal modalities for prediction and to visualize discriminative features within the signals.

Materials and methods

Data collection

We based our study on a physiological dataset from infants with Robin sequence. This group is well suited for automated prediction because of their homogeneous pathophysiology, that is, apneas and hypopneas are mainly caused by a narrow upper airway. Between May 2020 and April 2021, 19 infants with Robin sequence underwent whole night recordings in the Department of Neonatology at Tübingen University Hospital, using standard digital cardiorespiratory polysomnography (Remlogic, Natus Medical Incorporated, California, USA). Modalities recorded included nasal airflow obtained via a nasal pressure transducer and recorded at 200 Hz, as well as thoracic and abdominal respiratory efforts via respiratory inductance plethysmography recorded at 50 Hz. Furthermore, the heart was monitored via an electrocardiogram (EKG) and pulse plethysmogram (PPG) recorded at 200 and 100 Hz, respectively. The beat-to-beat heart rate was automatically derived from the EKG signal based on the RR-intervals. Finally, the SpO_2 and transcutaneous PCO_2 levels were recorded [19].

Five different types of adverse events were annotated in the recordings by a domain expert according to the criteria of the 2020 American Academy of Sleep Medicine (AASM) guidelines [19]. These were central apnea, obstructive apnea, and mixed apnea as well as central and obstructive hypopnea. Additionally, intermittent hypoxia events (desaturations) and body movements were annotated. Details about the annotation criteria can be found in S1 Appendix.

Prediction setup

We framed the problem of apnea and hypopnea prediction as a binary time-series classification problem. For this purpose, all types of apneas and hypopneas were combined into a single type of adverse event. No attempt was made to distinguish between different types of apnea and hypopnea. Isolated intermittent hypoxia events were not considered adverse events. Parts of the signal annotated as either adverse event or movement were removed from the signal. The remaining signal was then divided into 30-second non-overlapping time windows. If a time window preceded an adverse event by an offset of 15 seconds, it was labeled as a target time window. This offset was well above the maximum duration of annotated apneas and hypopneas, ensuring that our prediction was not based on imprecise annotation time stamps. If a time window was at least three minutes away from both the start and end of an adverse event, it was labeled as a control time window (Fig 1). The choice of 30-second time windows was not arbitrary: Kelly and Shannon [20] define periodic breathing as "three or more episodes

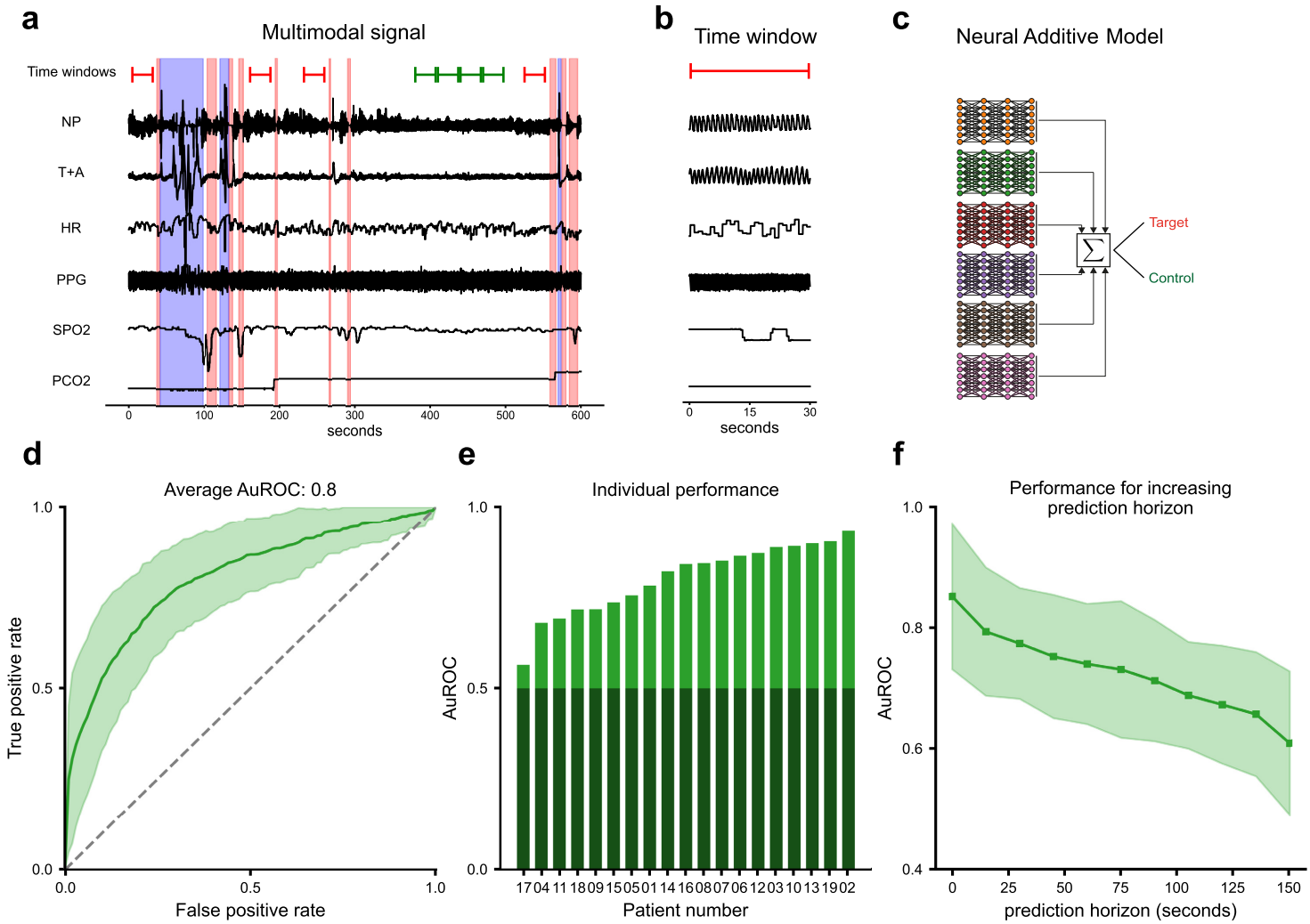


Fig 1. Overview of the setup and core results. a) 19 overnight respiratory polygraphy recordings of infants with Robin sequence were collected. Six signals were used for analysis: Nasal pressure (NP), the sum of thoracic and abdominal respiratory effort (T+A), heart rate (HR), photoplethysmogram (PPG), SpO_2, and PCO_2 levels. Adverse events (apneas and hypopneas, indicated as pink vertical bars) as well as infant movement (blue bars) were annotated manually. From these annotations, 30 second target time windows shortly preceding the adverse events (red), as well as control time windows (green) were extracted. **b)** Example of an extracted target time window. **c)** To discriminate between target and control time windows, a neural additive model (NAM) [18] was trained and tested with patient-based leave-one-out cross validation. **d)** Classification performance of the NAM: Pointwise average over the ROC curves of individual patients. Uncertainty corresponds to the pointwise standard deviation. **e)** AuROC for the individual patient-based test sets sorted in increasing order. Random performance is indicated in dark green. **f)** The average test performance as a function of an increasing prediction horizon. Uncertainty corresponds to the standard deviation over the performance of different patients.

https://doi.org/10.1371/journal.pdig.0000678.g001

of central apnea lasting at least 4 seconds, each separated by no more than 20 seconds of normal breathing". Therefore, this fixed length ensured a minimum distance between annotated adverse events to avoid the short inter-apnea episodes that occur during periodic breathing and are easily distinguished from normal breathing by their length alone [6, 21].

To evaluate the performance of the classifier, we performed patient-based leave-one-out cross-validation. That is, we used all extracted time windows of a single patient as a test set and performed the training and validation on the totality of all time windows extracted from the remaining patients. This procedure was repeated for each patient, and average performance scores and standard deviations are reported. For each iteration, we selected the hyperparameters using nested cross-validation [22]. This leave-one-out approach ensured that there was

no information leakage between the training and test sets, as the time windows on which the model was tested were extracted from a completely independent patient.

We focused on six signal modalities: Nasal pressure, the sum of thoracic and abdominal respiratory effort, heart rate (derived from EKG), PPG, SpO_2, and PCO_2 signals. Inter-breath intervals based on thoracic respiratory effort and heart rate have "classically" been used to predict apnea and hypopnea [6]. Since thoracic and abdominal respiratory effort are highly correlated signals, we treated them as a single signal by summing both signal traces pointwise. Creating the sum of these two modalities is a well-established practice in the analysis of apnea and hypopnea episodes [23, 24]. In addition, nasal pressure represents another interesting respiratory signal that may carry different information than the "classically" used thoracic and abdominal respiratory efforts. We also included the PPG signal and the two blood related parameters SpO_2 and PCO_2.

Preprocessing

The three respiratory signals, nasal pressure, and thoracic and abdominal respiratory effort, sampled at 200 or 50 Hz, respectively, were downsampled to 5 Hz after application of an order 8 Chebyshev filter [25]. The same preprocessing was applied to the PPG signal. Both the SpO_2 and PCO_2 signals were downsampled from 2 to 1 Hz. The derived heart rate was left unchanged at 1 Hz. We then standardized all respiratory signals and the PPG signal on a time window basis. The summed signal of thoracic and abdominal respiratory effort was obtained by summing both standardized signals pointwise and standardizing the result again. Heart rate, SpO_2, and PCO_2 were range-normalized from 50 to 240 bpm, 60 to 100%, and 30 to 70 mmHg to a range of -1 to 1. No further preprocessing, data cleaning or missing value imputation was performed.

Neural additive models for time series

Since there has been little prior work on possible features for individual apnea and hypopnea prediction [10], we performed automatic feature extraction with deep neural networks. To ensure interpretability, we used a neural additive model (NAM), which is a special case of a generalized additive model (GAM). GAMs combine expressiveness with built-in interpretability and have been used to analyze tabular data, especially in the medical domain [26–28]. For tabular data, GAMs are a linear combination of scalar, non-linear functions f_i:

$$\hat{p} = \sigma\left(\sum_i \alpha_i f_i(x_i) + \beta\right).$$

In binary classification, $\hat{p}$ denotes the classification probability and $\sigma(x) = (1 + e^x)^{-1}$ denotes the sigmoid function. GAMs have traditionally been fitted with splines or decision trees, but recent work has parameterized the functions f_i with neural networks, giving rise to NAMs [18, 29]. Both GAMs and NAMs are inherently interpretable because it is possible to visualize each f_i as a function of the corresponding feature. Moreover, given a specific data sample x_i, it is possible to analyze the individual importance of each feature by measuring the additive contribution $\alpha_i f_i(x_i)$ to the overall classification score.

Unlike GAMs, NAMs are not limited to tabular data. It is possible to choose f_i to be any type of neural network architecture and thus to input whole time series instead of tabular values. When using more general networks to input entire time series x_i instead of scalar features, it becomes difficult to visualize f_i. However, we can still measure the additive contribution of

each network $\alpha_i f_i(x_i)$ and thus still quantify the importance of a single time series modality to the overall classification.

To overcome the visualization difficulties, we parameterized the functions f_i by fully convolutional networks (FCNs) [30, 31]. This allowed us to visualize discriminative features in the time series with activation maps [32].

Architecture and training details

We implemented the presented NAM, as well as all downstream analyses in Python using the Pytorch [33] and the Scikit-learn [34] library. The individual subnetworks consisted of three convolutional layers. Kernel sizes were selected with nested cross-validation and ranged between 5 and 17. We used zero padding to preserve the time dimension of each input signal. Like the kernel size, the number of convolutional hidden channels was also selected with nested cross validation, which resulted in 20 hidden channels per layer. After each convolutional layer, we applied batch normalization [35] and rectified linear units (ReLUs) [36]. Before the global average pooling layer, network activations were linearly combined per time point to produce one single time series activation map for each signal modality.

The Adam optimizer [37, 38] was used to train the classifier with a standard binary cross entropy loss. To avoid training issues due to class imbalance, we undersampled the control time windows on a patient basis during training to balance control and target time windows. We trained the networks for 10 epochs with a learning rate of 0.0001 and a weight decay of 0.01. These choices resulted in optimal performance in every validation fold. The previously mentioned network hyperparameters were also stable across validation folds. This stability of hyperparameters across subjects was a consequence of our leave-one-out approach, where training sets differed only slightly between validation folds.

Comparison to baseline models

As baseline models against which to compare our NAM, we trained a logistic regression and a multi layer perceptron (MLP) classifier, which perform the prediction based on a total of 24 engineered features. We engineered six features for the higher-frequency oscillatory modalities (i.e. nasal pressure, thoraric and abdominal respiratory effort, photoplethysmogram): Skewness and kurtosis of the signal in the time domain, as well as the centroid, spread, skewness and kurtosis of the power spectrum. For the three lower-frequency modalities (heart rate, SpO_2 and PCO_2 levels), we computed two features, that is, the mean and range (maximum minus minimum value) of the given time window.

Additionally, to quantify any trade-off in terms of model interpretability and prediction performance, we compared our NAM to a blackbox neural network model that works directly on the time series. We built this blackbox model by passing the high-dimensional features extracted by the individual subnetworks trough a MLP. Like the NAM, we trained this blackbox architecture end-to-end. Details on the feature engineering and baselines can be found in S1 Appendix.

To assess the significance of potential differences between different models, we always repeated our training procedure ten times to obtain average performances for each infant and computed Wilcoxon signed-rank (p_{wil}) tests over the $n = 19$ patients. Furthermore, we performed permutation tests [39] to check whether models perform significantly better than random. See S1 Appendix for more details on the statistical tests.

Ethics

The ethics committee of the University Hospital Tübingen gave ethical approval for this work (application number: 352/2021BO2). The data was originally recorded for clinical purposes. The ethics committee of the University Hospital Tübingen gave permission to use and publish the anonymised data for research purposes without additional written informed consent from participants' parents.

Results

Descriptive statistics

We performed our analyses on a dataset of $n = 19$ infants with Robin sequence who were admitted to our Department of Neonatology between May 2020 and April 2021. Gestational age at birth was 39 (32–41) [median (range)] weeks, birth weight was 3,390 (1,320–4,380) g. At the time of respiratory polygraphy, infants were 17 (1–73) days old and weighed 3,392 (2,642–4,380) g. A total of 185 hours of respiratory polygraphy data were recorded, including 122 hours of total sleep time. Infants experienced 27 (3–112) obstructive, mixed and central apnea events per hour, and 28 (1–58) obstructive and mixed hypopnea events per hour, with events lasting for 3.8 (3.9–5.7) and 4.9 (2.8–7.3) seconds per event, respectively. After removing parts of the signals annotated as either adverse events or movements, the signals were divided into 30-second control and pre-adverse event time windows. This procedure resulted in 214 (75–606) time windows per infant to be used for classification. In all but one patient, there were more control time windows than target time windows. On average, 36% (4–66%) of windows corresponded to target events.

Performance of neonatal adverse event prediction

Our neural additive model predicts neonatal apnea and hypopnea by performing a classification into pre-adverse event and control time windows. The average AuROC over all leave-one-out test sets was 0.80 with a standard deviation of 0.093 (Fig 1d). For some patients, we achieved classification performance above 0.9 AuROC (Fig 1e). We performed permutation tests [39] to assess statistical significance of test performance. For all but one patient ($p_{per} = 0.121$) the performance was significantly better than random ($p_{per} < 0.001$, see S3 Table for all permutation test results). Additionally, we computed true positive rates and precision scores at fixed false positive rates (FPR). For a fixed FPR of 20%, the NAM achieved an average true positive rate and precision of 70% (Table 1, see S4 Table for the full per-patient confusion matrices at 20% FPR).

We also investigated how much the performance altered when the prediction horizon between the target time window and the adverse event is increased or decreased from its default value of 15 seconds. As expected, as the prediction horizon increased, the classification performance decreased and reached near chance level after about 150 seconds. On the other hand, for no offset between annotated events and target time windows, the performance increased (Fig 1f).

Table 1. True positive rate (TPR) and precision achieved by the NAM. Both metrics were computed at different levels of false positive rate (FPR) aggregated over all 19 patients [median (25–75% quantile)].

	at 10% FPR	at 20% FPR	at 30% FPR
TPR (%)	47 (33–66)	70 (51–79)	80 (60–87)
Precision (%)	76 (60–85)	70 (52–75)	61 (45–68)

https://doi.org/10.1371/journal.pdig.0000678.t001

Similarly, we investigated whether increasing the time window length would improve prediction performance. We tested this by doubling the time window length from 30 seconds to 60 seconds. We found that increasing the time window length did not significantly improve prediction performance (see S1 Appendix for details).

Our feature-based baselines, that is, the logistic regression and the multi-layer perceptron (MLP) classifier, also achieved a good average performance of 0.778 and 0.777, respectively. However, when comparing the interpretable NAM with the interpretable logistic regression, the NAM was significantly better ($p_{Wil} = 0.014$). Interestingly, the logistic regression, which linearly combines the 24 features was not worse than the MLP classifier, which is able to learn a complex, non-linear function of the same features. This is likely due to the fact that some of the information contained in the different modalities is redundant. We observed the same for the full blackbox neural network when compared to our NAM: The blackbox model was as good as our NAM, but did not achieve a significantly higher average performance ($p_{Wil} = 0.623$, S1 Table). In such cases, where many models achieve similar performance on a given problem, choosing inherently interpretable models is preferable [15–17]. See the Supporting information for additional information regarding the baseline model performances (S1 Fig and S1 Table).

Neural additive model for signal modality importance

Next, we investigated how much predictive information the six physiological signal modalities carry for the classification. Our NAM classifier allowed us to investigate this by analyzing the additive contribution of each signal modality to the overall classification score. The higher the absolute value of an individual additive contribution is compared to the other contributions, the more influence it has on the overall classification. To assess the overall importance of the six signal modalities used for classification, we analyzed the additive contributions pooled across all patient test sets (Fig 2a). To analyze differences between patients, we calculated the standard deviations over the additive contributions of individual patients (Fig 2b). To

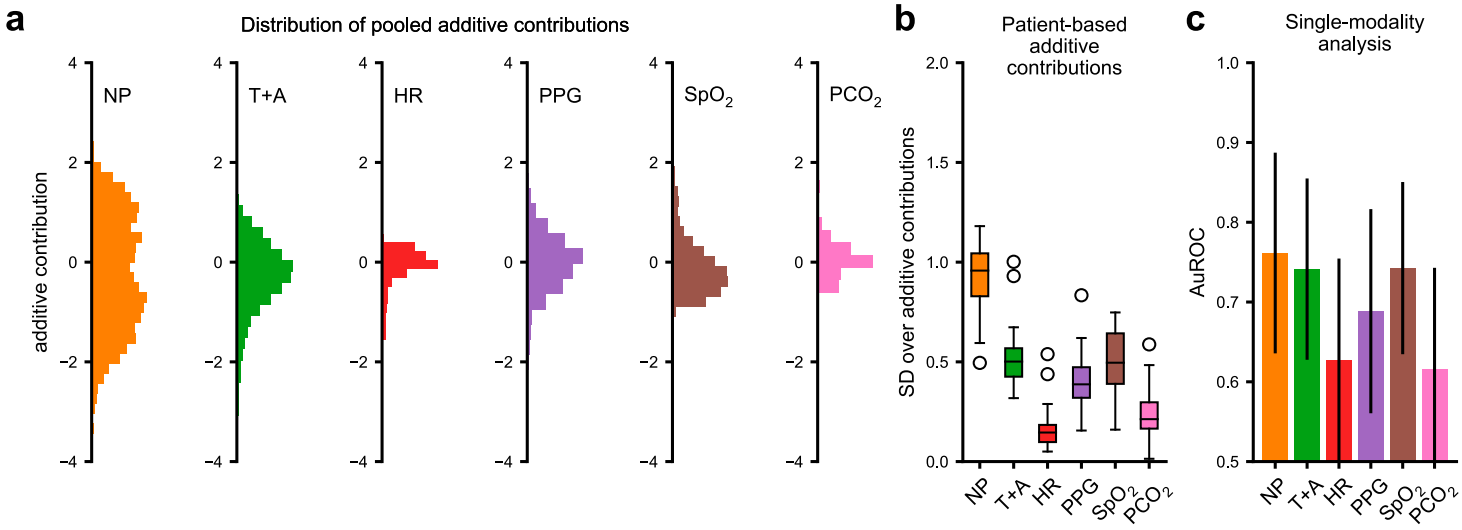


Fig 2. Importance analysis of the signal modalities. For the neural additive model (NAM), the importance of a signal modality for a given prediction can be measured by analyzing its associated additive contributions. **a)** Histograms of additive contributions pooled over all patient-based test sets (NP for nasal pressure, T+A for thoracic and abdominal respiratory effort, HR for heart rate, PPG for photoplethysmogram, and SpO_2 and PCO_2 levels). **b)** Box plots of the distribution over the standard deviation (SD) of additive contributions of all individual patients. **c)** Average AuROC scores of the individually trained single modality networks together with the standard deviation. All analyses show that the nasal pressure signal contains the most predictive information on average.

https://doi.org/10.1371/journal.pdig.0000678.g002

corroborate our analysis of the additive contributions, we also performed a single modality analysis, where the subnetwork of each signal modality is trained and evaluated independently.

The large standard deviations of both the pooled and patient-based additive contributions indicated that the nasal pressure signal contained the most predictive information (Fig 2a and 2b). The other respiratory signal, the sum of thoracic and abdominal respiratory effort, was comparatively less important for the prediction. This result was confirmed by the performance of the single modality networks: The nasal pressure network achieved the highest average AuROC, which was significantly better than the performance of the thoracic and abdominal effort network ($p_{wil} = 0.032$, Fig 2c). The importance of both respiratory signals was consistent with the clinical perspective, as infants with Robin sequence mainly suffer from obstructive apnea and hypopnea. Furthermore, both the NAM and the single modality analysis showed that the other modalities heart rate, PPG, PCO_2 and especially SpO_2 carried predictive information in addition to the respiratory signals (S2 and S3 Tables). The performance of the single modality SpO_2 was comparable to that of the nasal pressure signal (no significant difference: $p_{wil} = 0.241$). The power of SpO_2 was expected clinically: apneas and hypopneas often occur in clusters, and thus decreased SpO_2 values may indicate shortly preceding adverse events that increase the likelihood of another upcoming adverse event. A similar effect could be observed for heart rate, as apneas and hypopneas are often followed by bradycardia. In addition, the PPG signal is very sensitive to small movements, and thus can detect pre-apnea or pre-hypopnea arousal.

Although some of the signal modalities contained less information than others, the additive combination of their information increased the overall performance: The NAM achieved significantly better average AuROC than each of the single modality networks ($p_{wil} = 0.009$, S1 Table).

Based on this modality importance analysis, we trained a NAM using only the three most predictive signal modalities, that is, the nasal pressure, the sum of thoracic and abdominal respiratory effort, and the SpO_2 level. The "reduced" NAM achieved an average AuROC of 0.802 that was not significantly different from the full NAM using all six modalities (S1 Table). As discussed in the comparison with the blackbox model, this result suggests some redundancy in the predictive information between modalities.

Activation maps for local interpretability

While analysis of the additive contributions of the NAM and single modality performances provided insight into which of the signal modalities are relevant for model prediction, it did not provide information about which features of each signal modality were discriminative of an impending adverse event. For this purpose, we computed activation maps to visualize the discriminative features (Fig 3).

In our case, each time window classified by the NAM resulted in a total of six activation maps, one for each signal modality. Positive activations indicated that the corresponding segment of the signal was discriminative towards a target time window. In contrast, negative activations indicated that a segment of the signal was discriminative towards a control time window. Based on visual analysis, the classifier focused on irregularities in the respiratory signal for both nasal pressure and the sum of thoracic and abdominal respiratory efforts. In addition to these features, the classifier used arousals on the plethysmograph, decreased SpO_2 levels or SpO_2 desaturations, and heart rate variations to make a classification.

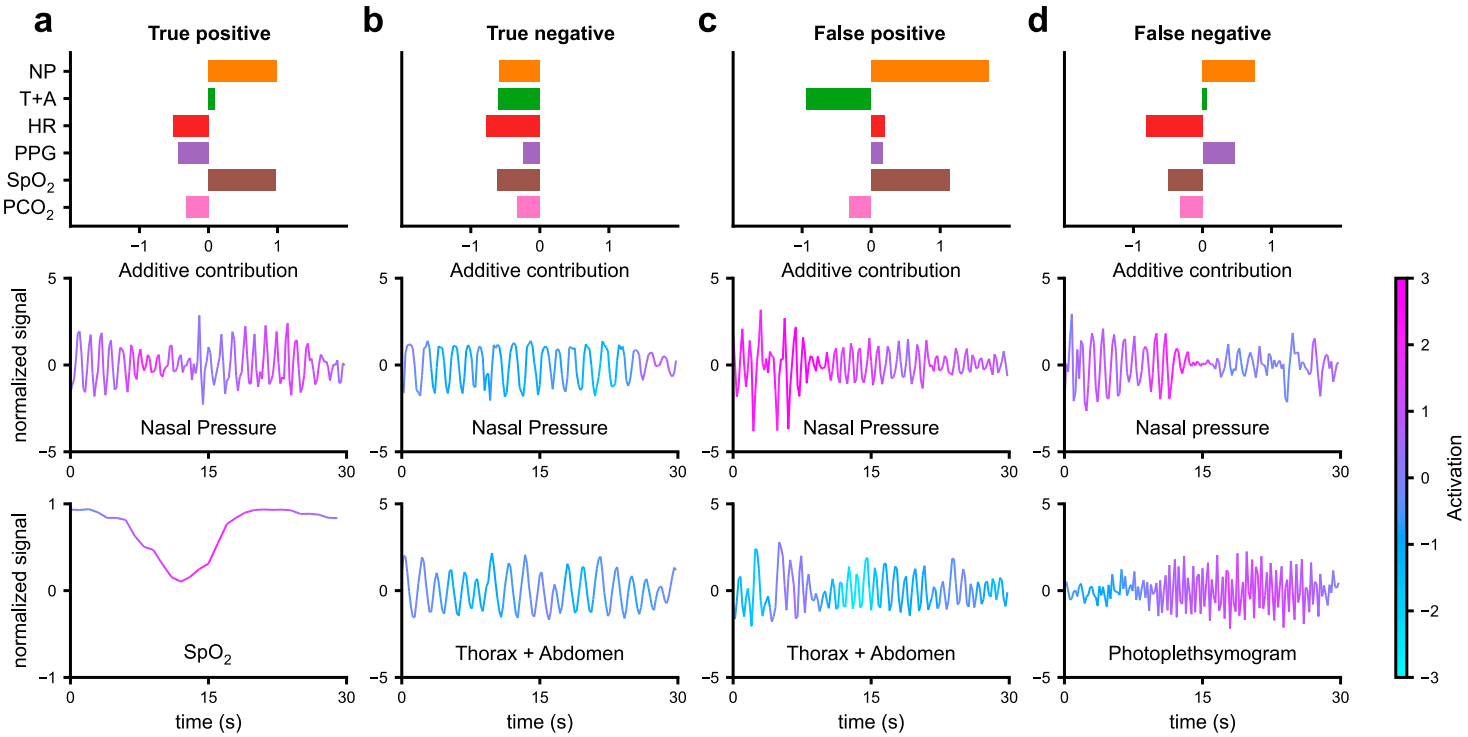


Fig 3. Exemplary activation maps together with the associated additive contributions. NP for nasal pressure, T+A for thoracic and abdominal respiratory effort, HR for heart rate, PPG for photoplethysmogram, and SpO_2 and PCO_2 levels. **a)** Correctly classified target (pre-adverse-event) time window. Light blue indicates negative activation and a classification towards no adverse event. Pink indicates positive activation and a classification towards an upcoming adverse event. The classifier detected different types of irregularities in the breathing signal as well as variations in heart rate and SpO_2 levels. **b)** Correctly classified control time window **c)** Incorrectly classified control time window (false positive). **d)** Incorrectly classified target time window (false negative).

https://doi.org/10.1371/journal.pdig.0000678.g003

Discussion

We developed an interpretable time-series classifier and applied it to the problem of predicting neonatal apnea and hypopnea in infants with upper airway obstruction. In contrast to previous work, we focused on the prediction of single apnea and hypopnea events. Our neural additive model (NAM) was able to automatically extract relevant features from the multimodal polygraphy signal and achieved good performance in classifying individual time windows. Importantly, because of the inherent interpretability of the NAM, we were able to perform several downstream analyses to gain insight into the features used by the model.

First, we investigated the importance of each signal modality recorded with respiratory polygraphy. Instead of the classically used thoracic (and abdominal) respiratory effort [6], nasal pressure proved to be more informative in predicting individual events. This effect could be observed both in the additive contributions of the NAM and the performance of the single-modality networks. From a practical point of view, the predictive power of nasal pressure may allow clinicians to reduce the burden of respiratory polygraphy. If full respiratory polygraphy is not required clinically, monitoring nasal pressure alone may be sufficient to predict apneas and hypopneas. Alternatively, a NAM using only a subset of the modalities could be used. Second, we were able to visualize the learned features using activation maps. These visualizations allowed us to confirm their clinical significance.

The explanations provided by our model (additive contributions and activation maps) could play an important role in future clinical applications where nurses and NICU staff interact with automated apnea and hypopnea prediction systems. Unlike interpretable models

based on potentially complicated and unintuitive features (such as for example the spectral moments used in the logistic regression baseline), the explanations provided by the NAM add to the existing polysomnography recording to which practitioners are accustomed: The additive contributions show which of the recorded modalities were relevant to the prediction made, and the activation maps highlight the areas within the recording that were considered discriminative by the model. Based on these explanations, nurses and NICU staff have the capability to decide whether to trust or distrust the model's prediction. In the case of systematic false alarms, NICU staff could then intervene by recalibrating the model or, in extreme cases, turning it off altogether. Allowing for this type of distrust in a model is critical for use in real-world clinical settings [15].

The fact that the NAM works directly on minimally pre-processed polysomnography recordings also has computational advantages. In an automated system, the prediction itself could be realized almost instantaneously, leaving enough time to trigger potential interventions. In our study, we consider a prediction window of 15 seconds, which would be enough time to trigger interventions such as sensory stimulation [40, 41].

Limitations

Although this study represents a first step towards the prediction of individual apnea and hypopnea events in neonates using machine learning, there are several obstacles that would need to be overcome in future work to allow the application of the presented model in a clinical setting. For some patients, the prediction performance was not sufficient to be of practical use. Especially for systems that perform automated interventions, high prediction accuracy will be necessary, as errors in a clinical setting may be costly: A warning system that produces many false negatives would not significantly reduce the stress an infant experiences from repeated apneas and hypopneas. Conversely, a warning system that produces many false positives would potentially place additional stress on the infant by prompting unnecessary interventions. Additional measures would also be needed to ensure the robustness of the model to unseen data, outliers, and malfunctioning recording devices to enable safe use.

Furthermore, in this study we were only concerned with performing "post-hoc" predictions of apneas and hypopneas. In a true interventional setting, the distribution of adverse events is subject to a distribution shift that may affect the performance of our current prediction model. We also excluded any periods of infant movement from analysis that would need to be dealt with in a real-world application. As such, this study represents a first step toward automated systems, but the practical validity of our model requires further study. Because our used dataset consisted of only 19 infants with Robin sequence who have a homogeneous pathophysiology, future studies need to apply the NAM to larger datasets and datasets collected in other settings with different patient demographics and more heterogeneous pathophysiology to provide external validation of our approach.

Finally, while the use of such automated machine learning systems in neonatal intensive care and health care in general is promising, the use of such systems raises challenging ethical considerations regarding (parental) consent, privacy, and accountability for potential harm.

Conclusion

In this study, we developed a neural additive model that predicts individual events of neonatal apnea and hypopnea in infants with Robin sequence. Despite its limitations, our neural additive model represents a step towards automated prediction of neonatal apnea and hypopnea. If

reliable, such a model would have the potential to optimize the care of vulnerable neonates with upper airway obstruction.

Supporting information

S1 Fig. Comparison of neural additive model (NAM) to baselines models.
(PDF)

S1 Appendix. Details on the significance analysis, the baselines and feature engineering, as well as the description and definition of adverse events.
(PDF)

S1 Table. NAM vs. single modality networks vs. baselines.
(PDF)

S2 Table. Statistical test results for single modality network performances.
(PDF)

S3 Table. Permutation tests for both the NAM and all single modality networks.
(PDF)

S4 Table. Confusion matrices for individual patients.
(PDF)

Acknowledgments

We would like to thank the staff, parents and infants involved in this study, Ms. Gabriele Hilber-Moessner who provided additional support in obtaining the data, and Dr. Mirja Quante who provided valuable advice with annotating the data. We also thank the International Max Planck Research School for Intelligent Systems (IMPRS-IS) and the AI4Med-BW graduate program for supporting J.V.

Author Contributions

Conceptualization: Julius Vetter, Jakob H. Macke, Christian F. Poets.

Data curation: Kathleen Lim, Christian F. Poets.

Formal analysis: Julius Vetter.

Funding acquisition: Oliver Kohlbacher, Jakob H. Macke, Christian F. Poets.

Investigation: Julius Vetter.

Methodology: Julius Vetter.

Software: Julius Vetter.

Supervision: Tjeerd M. H. Dijkstra, Peter A. Dargaville, Oliver Kohlbacher, Jakob H. Macke, Christian F. Poets.

Validation: Julius Vetter.

Visualization: Julius Vetter.

Writing – original draft: Julius Vetter.

Writing – review & editing: Julius Vetter, Kathleen Lim, Tjeerd M. H. Dijkstra, Peter A. Dargaville, Oliver Kohlbacher, Jakob H. Macke, Christian F. Poets.

References

1. Bloch-Salisbury E, Indic P, Bednarek F, Paydarfar D. Stabilizing immature breathing patterns of preterm infants using stochastic mechanosensory stimulation. Journal of Applied Physiology. 2009; 107 (4):1017–1027. https://doi.org/10.1152/japplphysiol.00058.2009 PMID: 19608934
2. Cattani L, Alinovi D, Ferrari G, Raheli R, Pavlidis E, Spagnoli C, et al. Monitoring infants by automatic video processing: A unified approach to motion analysis. Computers in Biology and Medicine. 2017; 80:158–165. https://doi.org/10.1016/j.compbiomed.2016.11.010 PMID: 27940321
3. Vergales BD, Paget-Brown AO, Lee H, Guin LE, Smoot TJ, Rusin CG, et al. Accurate automated apnea analysis in preterm infants. American journal of Perinatology. 2014; 31(2):157–162. https://doi.org/10.1055/s-0033-1343769 PMID: 23592319
4. Clark MT, Rusin CG, Hudson JL, Lee H, Delos JB, Guin LE, et al. Breath-by-breath analysis of cardio-respiratory interaction for quantifying developmental maturity in premature infants. Journal of Applied Physiology. 2012; 112(5):859–867. https://doi.org/10.1152/japplphysiol.01152.2011 PMID: 22174403
5. Altuve M, Carrault G, Beuchee A, Pladys P, Hernández AI. Online apnea-bradycardia detection using hidden semi-Markov models. In: Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE; 2011. p. 4374–4377.
6. Zuzarte I, Sternad D, Paydarfar D. Predicting apneic events in preterm infants using cardio-respiratory and movement features. Computer Methods and Programs in Biomedicine. 2021; 209:106321. https://doi.org/10.1016/j.cmpb.2021.106321 PMID: 34380078
7. Pravisani G, Beuchee A, Mainardi L, Carrault G. Short term prediction of severe bradycardia in premature newborns. In: Computers in Cardiology. IEEE; 2003. p. 725–728.
8. Mahmud MS, Wang H, Kim Y. Accelerated prediction of bradycardia in preterm infants using time-frequency analysis. In: International Conference on Computing, Networking and Communications (ICNC). IEEE; 2019. p. 468–472.
9. Shirwaikar RD, Acharya D, Makkithaya K, Surulivelrajan M, Srivastava S. Optimizing neural networks for medical data sets: A case study on neonatal apnea prediction. Artificial intelligence in medicine. 2019; 98:59–76. https://doi.org/10.1016/j.artmed.2019.07.008 PMID: 31521253
10. Lim K, Jiang H, Marshall AP, Salmon B, Gale TJ, Dargaville PA. Predicting apnoeic events in preterm infants. In: Frontiers in Pediatrics; 2020. p. 570.
11. Williamson JR, Bliss DW, Paydarfar D. Forecasting respiratory collapse: theory and practice for averting life-threatening infant apneas. Respiratory physiology & neurobiology. 2013; 189:223–231. https://doi.org/10.1016/j.resp.2013.05.034 PMID: 23735485
12. Ismail Fawaz H, Forestier G, Weber J, Idoumghar L, Muller PA. Deep learning for time series classification: A review. Data mining and knowledge discovery. 2019; 33(4):917–963. https://doi.org/10.1007/s10618-019-00619-1
13. Futoma J, Hariharan S, Heller K, Sendak M, Brajer N, Clement M, et al. An improved multi-output gaussian process RNN with real-time validation for early sepsis detection. In: Machine Learning for Healthcare Conference. PMLR; 2017. p. 243–254.
14. Lea C, Flynn MD, Vidal R, Reiter A, Hager GD. Temporal convolutional networks for action segmentation and detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition; 2017. p. 156–165.
15. Rudin C. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nature Machine Intelligence. 2019; 1(5):206–215. https://doi.org/10.1038/s42256-019-0048-x PMID: 35603010
16. Rudin C. Why black box machine learning should be avoided for high-stakes decisions, in brief. Nature Reviews Methods Primers. 2022; 2(1):81. https://doi.org/10.1038/s43586-022-00172-0
17. Semenova L, Rudin C, Parr R. On the existence of simpler machine learning models. In: 2022 ACM Conference on Fairness, Accountability, and Transparency; 2022. p. 1827–1858.
18. Agarwal R, Melnick L, Frosst N, Zhang X, Lengerich B, Caruana R, et al. Neural additive models: Interpretable machine learning with neural nets. In: Advances in Neural Information Processing Systems. vol. 34; 2021. p. 4699–4711.
19. Lim K, Quante M, Dijkstra T, Hilbert-Moessner G, Wiechers C, Dargaville P, et al. Should obstructive hypopneas be included when analyzing sleep studies in infants with Robin Sequence? Sleep Medicine. 2022; 98:9–12. https://doi.org/10.1016/j.sleep.2022.06.010 PMID: 35764010
20. Kelly DH, Shannon DC. Treatment of apnea and excessive periodic breathing in the full-term infant. Pediatrics. 1981; 68(2):183–186. https://doi.org/10.1542/peds.68.2.183 PMID: 7267223

21. Richards JM, Alexander JR, Shinebourne EA, De Swiet M, Wilson AJ, Southall DP. Sequential 22-hour profiles of breathing patterns and heart rate in 110 full-term infants during their first 6 months of life. Pediatrics. 1984; 74(5):763–777. https://doi.org/10.1542/peds.74.5.763 PMID: 6238275

22. Cawley GC, Talbot NLC. On Over-fitting in Model Selection and Subsequent Selection Bias in Performance Evaluation. Journal of Machine Learning Research. 2010; 11(70):2079–2107.

23. Esquer C, Claure N, D'Ugard C, Wada Y, Bancalari E. Mechanisms of hypoxemia episodes in spontaneously breathing preterm infants after mechanical ventilation. Neonatology. 2008; 94(2):100–104. https://doi.org/10.1159/000116634 PMID: 18277057

24. Di Fiore JM. Neonatal cardiorespiratory monitoring techniques. Seminars in Neonatology. 2004; 9 (3):195–203. https://doi.org/10.1016/j.siny.2003.11.009 PMID: 15050212

25. Williams AB, Taylor FJ. Electronic filter design handbook. McGraw-Hill Education; 2006.

26. Hastie TJ. In: Generalized Additive Models. Routledge; 2017. p. 249–307.

27. Lou Y, Caruana R, Gehrke J, Hooker G. Accurate intelligible models with pairwise interactions. In: Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining; 2013. p. 623–631.

28. Lou Y, Caruana R, Gehrke J. Intelligible models for classification and regression. In: Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining; 2012. p. 150–158.

29. Srivastava D, Aydin B, Mazzoni EO, Mahony S. An interpretable bimodal neural network characterizes the sequence and preexisting chromatin predictors of induced transcription factor binding. Genome biology. 2021; 22(1):1–25. https://doi.org/10.1186/s13059-020-02218-6 PMID: 33413545

30. Gu J, Wang Z, Kuen J, Ma L, Shahroudy A, Shuai B, et al. Recent advances in convolutional neural networks. Pattern recognition. 2018; 77:354–377. https://doi.org/10.1016/j.patcog.2017.10.013

31. Wang Z, Yan W, Oates T. Time series classification from scratch with deep neural networks: A strong baseline. In: 2017 International joint conference on neural networks (IJCNN). IEEE; 2017. p. 1578–1585.

32. Zhou B, Khosla A, Lapedriza A, Oliva A, Torralba A. Learning deep features for discriminative localization. In: Proceedings of the IEEE conference on computer vision and pattern recognition; 2016. p. 2921–2929.

33. Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, et al. Pytorch: An imperative style, high-performance deep learning library. In: Advances in neural information processing systems. vol. 32; 2019.

34. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research. 2011; 12:2825–2830.

35. Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International conference on machine learning. PMLR; 2015. p. 448–456.

36. Arora R, Basu A, Mianjy P, Mukherjee A. Understanding deep neural networks with rectified linear units. In: arXiv preprint arXiv:1611.01491; 2016.

37. Kingma DP, Ba J. Adam: A method for stochastic optimization. In: arXiv preprint arXiv:1412.6980; 2014.

38. Loshchilov I, Hutter F, et al. Fixing weight decay regularization in Adam. arXiv preprint arXiv:171105101. 2017;5.

39. Ojala M, Garriga GC. Permutation Tests for Studying Classifier Performance. Journal of Machine Learning Research. 2010; 11(6):1833–1863.

40. Lim K, Cramer SJ, Te Pas AB, Gale TJ, Dargaville PA. Sensory stimulation for apnoea mitigation in preterm infants. Pediatric Research. 2022; 92(3):637–646. https://doi.org/10.1038/s41390-021-01828-5 PMID: 34819656

41. Cramer SJ, Dekker J, Dankelman J, Pauws SC, Hooper SB, Te Pas AB. Effect of tactile stimulation on termination and prevention of apnea of prematurity: a systematic review. Frontiers in pediatrics. 2018; 6:45. https://doi.org/10.3389/fped.2018.00045 PMID: 29552548

Appendix

Details on the significance analysis

For a rigorous performance comparison between the Neural Additive Model (NAM) and single modality networks, we ran the complete training and testing pipeline ten times and compute the average AuROC and standard deviation over all patients. We then performed a significance analysis over the differences in average performance between the NAM and the single modality networks as well as the baseline models. To this end, we computed Wilcoxon signed-rank tests over the $n = 19$ patients. We further extended this significance analysis by comparing all pairs of single-modality network performances, again using Wilcoxon signed-rank tests. We also performed permutation permutation tests [1] for all patients for both the NAM and all single modality networks. See tables in supporting information for all results.

Details on baselines and feature engineering

Feature-based baselines To compare our NAM with more classical approaches, we trained a logistic regression and a multi-layer perceptron (MLP) classifier using the same leave-one-out approach. To make these baselines competitive, we computed a total of 24 features across the six signal modalities. For heart rate, SpO_2, and PCO_2, the average and range (i.e., maximum minus minimum) were computed over each 30-second time window to capture decelerations or abnormally low or high values. For the oscillatory modalities, that is, the nasal pressure, the thoracic and abdominal respiratory efforts, and the pulse plethysmogram, we computed (higher-order) moments in time and spectral domain to capture differences in the "regularity" of the oscillatory dynamics. More specifically, in the time domain, we computed the third and fourth central moment (skewness and kurtosis). In the spectral domain, we computed moments based on the spectral centroid (SC). The SC is given by

$$\mathrm{SC} = \frac{\sum_{n=0}^{N-1} f(n) \cdot X(n)}{\sum_{n=0}^{N-1} X(n)},$$

where $f(n)$ represents the frequency at bin n, $X(n)$ represents the magnitude (or power) of the signal at frequency bin n, and N is the total number of frequency bins. Based on the spectral centroid, we also computed the spectral spread, skewness and kurtosis (see [2] for details).

Blackbox neural network To compare our NAM to an uninterpretable but powerful model, we employed the individual single-modality networks used in the NAM to extract features into a high-dimensional latent space (20 dimensions per modality and 120 dimensions in total). This high-dimensional latent embedding is then passed through an MLP to perform classification. We trained both the single modality networks and the MLP end-to-end in a single architecture. This architecture is more powerful and expressive than the NAM, but the ability to compute additive contributions or visualize activations per modality is lost.

Training of baseline models We trained all baseline models using the identical leave-one-out approach as described for the NAM. Hyperparameters were chosen independently for each baseline model. For the MLP classifier and blackbox neural network, we again used the Adam optimizer [3, 4] with a learning rate of 0.0001 and a weight decay of 0.01, and trained for 10 epochs. For the logistic regression, we used the L-BFGS optimizer [5] with a learning rate of 0.01, a history size of 10, a L_2-regularization strength of 0.001 and trained for 100 iterations.

Different time window lengths To investigate whether longer prediction windows further improve prediction, we conducted an experiment with 60-second time windows. To ensure a fair comparison with 30-second time windows, we matched 30 and 60-second time windows one-to-one. This was achieved by first extracting 60-second time windows and then removing the second half to obtain the matching 30-second time windows. As a result, the 30-second time windows used in this experiment differ from those used in the main text.

We then trained the NAM on both time window lengths using the same hyperparameters as in the main experiment. The NAM with 60-second time windows achieved an average AuROC of 0.762, and the NAM with

30-second time windows achieved an AuROC of 0.764. Thus, there was no improvement with increasing time window length. This result is consistent with our earlier experiment, which showed that prediction performance deteriorates significantly with increasing prediction horizon. Clinically, the occurrence of irregularities in the polysomnography recording is expected to occur close to the onset of an apnea or hypopnea.

Description and definition of adverse events

The criteria used to annotate the recorded polygraphy signals are given in the following. More details can be found in [6].

1. Obstructive apnea
   - Lasts for ≥ 2 breaths with respiratory effort present on inductance belt
2. Central apnea
   - Lasts for ≥ 2 breaths with associated SpO_2 decrease by $\geq 3\%$
   - Lasts for ≥ 20 seconds
   - Arousal *or* heart rate < 50 bpm for ≥ 5 seconds *or* heart rate < 60 bpm for > 15 seconds
3. Mixed apnea
   - Lasts for ≥ 2 breaths with component of both obstructive and central apnea
4. Obstructive hypopnea
   - Nasal flow $\leq 70\%$ of baseline for ≥ 2 breaths with associated SpO_2 decrease by $\geq 3\%$
   - Arousal
   - Hypopnea with increased inspiratory flattening
   - Thoracoabdominal paradox
5. Central hypopnea
   - Without above-mentioned features of obstruction
6. Hypoxia
   - SpO_2 decrease by $\geq 3\%$ within a 5 seconds duration
7. Movement
   - Gross movement observed for at least 15 seconds
   - Eye opening
   - Two episodes of movement need to be separated by at least 15 seconds

Data and code availability

The dataset analyzed in this study is available on Zenodo: `https://zenodo.org/record/7711137`. The code for all experiments is available at `https://github.com/mackelab/neonatal_apnea_prediction`.

References

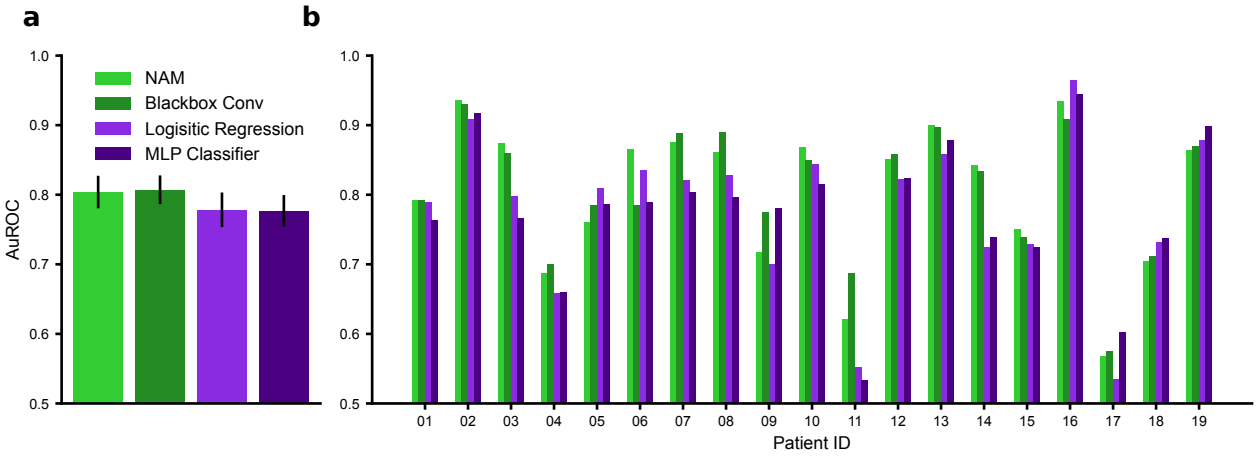
1. Ojala M, Garriga GC. Permutation Tests for Studying Classifier Performance. Journal of Machine Learning Research. 2010;11(6):1833–1863.
2. Peeters G, Giordano BL, Susini P, Misdariis N, McAdams S. The timbre toolbox: Extracting audio descriptors from musical signals. The Journal of the Acoustical Society of America. 2011;130(5):2902–2916.

3. Kingma DP, Ba J. Adam: A method for stochastic optimization. In: arXiv preprint arXiv:1412.6980; 2014.

4. Loshchilov I, Hutter F, et al. Fixing weight decay regularization in Adam. arXiv preprint arXiv:171105101. 2017;5.

5. Liu DC, Nocedal J. On the limited memory BFGS method for large scale optimization. Mathematical programming. 1989;45(1):503–528.

6. Lim K, Quante M, Dijkstra T, Hilbert-Moessner G, Wiechers C, Dargaville P, et al. Should obstructive hypopneas be included when analyzing sleep studies in infants with Robin Sequence? Sleep Medicine. 2022;98:9–12.



S1 Figure. Comparison of neural additive model (NAM) to baselines models. **a)** Average AuROC over all 19 infants of the neural additive model compared to the blackbox neural network as well as the feature-based logistic regression and multi-layer perceptron classifier. Error bars indicate standard error of the mean over the 19 infants. **b)** Performance of the four models for the 19 individual infants.

S1 Table. NAM vs. single modality networks vs. baselines. Average AuROC and standard deviation across different runs (Seed-SD) of the neural additive model (NAM), signal modality networks (with NP for nasal pressure, T+A for thoracic and abdominal respiratory effort, HR for heart rate, PPG for photoplethysmogram, and SpO_2 and PCO_2 levels), and baseline models (blackbox neural network, logistic regression, multi-layer perceptron (MLP) classifier) over 10 independent training and testing runs. The Wilcoxon tests are computed over the $n = 19$ patients with the AuROC performances for each patient averaged over all 10 runs. All tests are two-sided.

Model	AuROC	Seed-SD	Wilcoxon
NAM	0.803	0.0077	-
NP	0.753	0.0067	$= 0.009$
T+A	0.735	0.0042	< 0.001
SpO_2	0.733	0.0058	$= 0.006$
PPG	0.694	0.0074	< 0.001
HR	0.627	0.0073	< 0.001
PCO_2	0.614	0.0077	< 0.001
Blackbox	0.807	0.0078	$= 0.623$
Log. Reg.	0.778	0.0016	$= 0.014$
MLP	0.777	0.0027	$= 0.072$
Reduced NAM	0.802	0.0045	$= 0.859$

S2 Table. Statistical test results for single modality network performances. The Wilcoxon tests are computed over the $n = 19$ patients where the AuROC performance for each patient is the average over all 10 runs. All tests are two-sided. NP for nasal pressure, T+A for thoracic and abdominal respiratory effort, HR for heart rate, PPG for photoplethysmogram, and SpO_2 and PCO_2 levels.

Model	NP	T+A	SpO_2,	PPG	HR
T+A	$= 0.032$				
SpO_2	$= 0.241$	$= 0.829$			
PPG	$= 0.011$	$= 0.087$	$= 0.568$		
HR	$= 0.007$	$= 0.016$	$= 0.005$	$= 0.169$	
PCO_2	$= 0.012$	$= 0.018$	$= 0.003$	$= 0.123$	$= 0.49$

S3 Table. Permutation tests for both the NAM and all single modality networks. The permutation tests were performed with 1024 permutations. Exact p-values are given for values larger than 0.001. NP for nasal pressure, T+A for thoracic and abdominal respiratory effort, HR for heart rate, PPG for photoplethysmogram, and SpO_2 and PCO_2 levels.

ID	NAM	NP	T+A	SpO_2	PPG	HR	PCO_2
01	0.001	0.001	0.001	0.001	0.001	0.001	0.017
02	0.001	0.001	0.001	0.001	0.001	0.001	0.001
03	0.001	0.001	0.002	0.001	0.001	0.001	0.001
04	0.001	0.001	0.005	0.062	0.008	0.002	0.001
05	0.001	0.001	0.001	0.001	0.008	0.374	0.983
06	0.001	0.001	0.001	0.001	0.001	0.607	0.001
07	0.001	0.001	0.001	0.001	0.001	0.044	0.921
08	0.001	0.001	0.001	0.001	0.001	0.037	0.998
09	0.001	0.03	0.02	0.001	0.2	0.027	0.005
10	0.001	0.001	0.001	0.001	0.001	0.997	0.7
11	0.004	0.845	0.001	0.001	0.727	0.001	0.003
12	0.001	0.001	0.001	0.001	0.008	0.001	0.001
13	0.001	0.001	0.001	0.001	0.001	0.001	0.001
14	0.001	0.001	0.001	0.001	0.001	0.001	0.07
15	0.001	0.001	0.001	0.006	0.008	0.016	0.006
16	0.001	0.001	0.001	0.001	0.001	0.366	0.001
17	0.121	0.157	0.558	0.713	0.103	0.031	0.004
18	0.001	0.001	0.001	0.001	0.001	0.853	0.778
19	0.001	0.001	0.014	0.001	0.661	0.001	0.001

S4 Table. Confusion matrices for individual patients. Decision threshold set to achieve 20% false positive rate (FPR). TN for true negatives, FP for false positives, FN for false negatives, and TP for true positives.

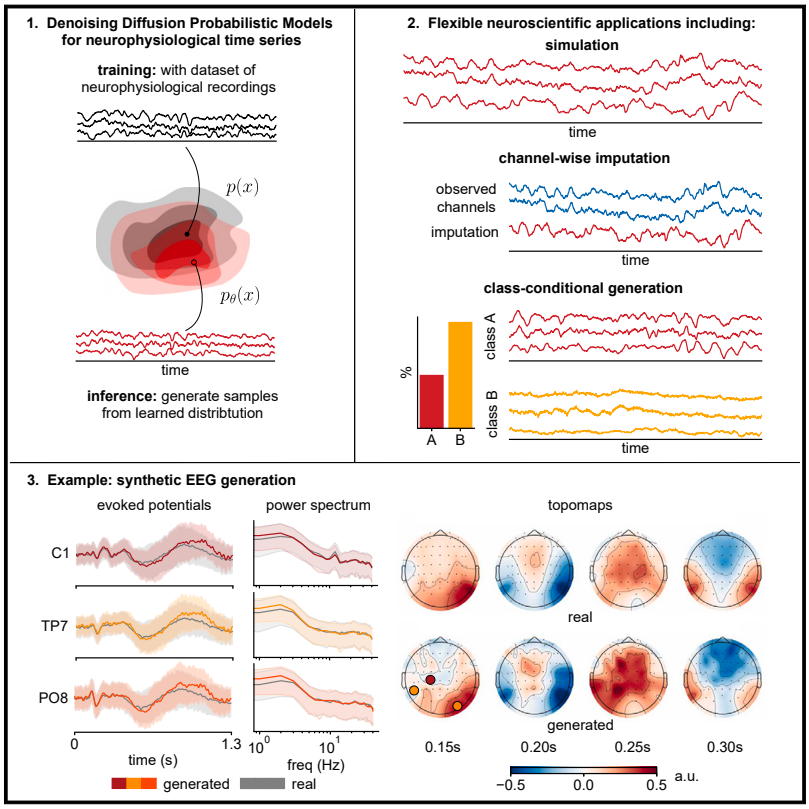
ID	TN	FP	FN	TP
01	123	30	58	66
02	122	22	9	61
03	62	15	10	35
04	215	54	36	29
05	24	5	26	31
06	197	50	15	55
07	137	32	21	73
08	109	26	15	60
09	45	10	34	36
10	149	36	13	43
11	33	7	19	16
12	81	20	15	62
13	170	43	10	49
14	278	71	20	46
15	424	97	14	14
16	44	10	23	43
17	468	114	14	10
18	111	26	34	27
19	46	8	7	31

Article

Patterns

Generating realistic neurophysiological time series with denoising diffusion probabilistic models

Graphical abstract



Highlights

- We present a denoising diffusion-based deep generative model for brain signals
- Our model accurately generates different data types, including EEG, ECoG, and LFP
- Data can also be generated while considering additional information, such as brain state
- Trained models and synthetic data facilitate several neuroscientific applications

Authors

Julius Vetter, Jakob H. Macke, Richard Gao

Correspondence

julius.vetter@uni-tuebingen.de (J.V.), jakob.macke@uni-tuebingen.de (J.H.M.), r.dg.gao@gmail.com (R.G.)

In brief

This study presents a denoising diffusion model tailored to neurophysiological time series. The model is able to generate a variety of neurophysiological time series, including electroencephalography (EEG), electrocorticography (ECoG), and local field potential (LFP) recordings, with high fidelity. The generated data accurately capture statistics of interest in neuroscience. In addition, the diffusion model can incorporate additional information into the generation process, facilitating several scientific and clinical applications, such as missing-data imputation or brain-state classification.

Vetter et al., 2024, Patterns 5, 101047
September 13, 2024
https://doi.org/10.1016/j.patter.2024.101047

CellPress

Patterns

CellPress
OPEN ACCESS

Article

Generating realistic neurophysiological time series with denoising diffusion probabilistic models

Julius Vetter,[1,4,*] Jakob H. Macke,[1,2,3,*] and Richard Gao[1,3,*]
[1]Machine Learning in Science, University of Tübingen and Tübingen AI Center, Tübingen, Germany
[2]Max Planck Institute for Intelligent Systems, Tübingen, Germany
[3]These authors contributed equally
[4]Lead contact
*Correspondence: julius.vetter@uni-tuebingen.de (J.V.), jakob.macke@uni-tuebingen.de (J.H.M.), r.dg.gao@gmail.com (R.G.)
https://doi.org/10.1016/j.patter.2024.101047

THE BIGGER PICTURE Diffusion models are a class of machine learning models that can generate high-fidelity synthetic data. In recent years, they have had a profound impact on engineering and science, revolutionizing image and audio synthesis and advancing our ability to model scientific data. Capitalizing on this, we have designed denoising diffusion probabilistic models (DDPMs) for modeling brain recordings, such as electroencephalography (EEG), electrocorticography (ECoG), and local field potential (LFP) recordings. The ability to accurately generate realistic neurophysiological data has many applications in experimental, clinical, and computational neuroscience, including simulation, missing-data imputation, brain-state classification, and more, which we demonstrate in the present study.

SUMMARY

Denoising diffusion probabilistic models (DDPMs) have recently been shown to accurately generate complicated data such as images, audio, or time series. Experimental and clinical neuroscience also stand to benefit from this progress, as the accurate generation of neurophysiological time series can enable or improve many neuroscientific applications. Here, we present a flexible DDPM-based method for modeling multichannel, densely sampled neurophysiological recordings. DDPMs can generate realistic synthetic data for a variety of datasets from different species and recording techniques. The generated data capture important statistics, such as frequency spectra and phase-amplitude coupling, as well as fine-grained features such as sharp wave ripples. Furthermore, data can be generated based on additional information such as experimental conditions. We demonstrate the flexibility of DDPMs in several applications, including brain-state classification and missing-data imputation. In summary, DDPMs can serve as accurate generative models of neurophysiological recordings and have broad utility in the probabilistic generation of synthetic recordings for neuroscientific applications.

INTRODUCTION

Statistical models that can accurately reconstruct complex datasets are useful in many areas of science. These so-called generative models allow scientists to interpret and analyze statistical dependencies between data features, as well as between data and mechanistic latent variables. Such models further grant the ability to generate realistic synthetic data, which are valuable in their own right, especially when it is possible to incorporate (or "condition on") additional information or observations into the model. Examples of applications include imputing missing data given observed data, forecasting the future given past data, creating synthetic datasets as input to a simulator, and replacing or augmenting scarce training data for subsequent machine learning models—all of which rely on a (conditional) generative model that can produce realistic synthetic data.

The generation of synthetic neurophysiological recordings has received much attention over the years, predominantly with a focus on generating spike trains with a prespecified correlation structure.[1–3] More generally, time series surrogate methods[4,5] can be used to generate synthetic neurophysiological recordings that mimic the real data in a well-defined set of features but lose complex, nonlinear features by design. Other works have employed deep neural networks as nonlinear encoding models of

brain response, for example, predicting electroencephalography (EEG)[6] response from viewed images. However, they are deterministic by design and cannot be used to probabilistically simulate neural time series, nor can they be used without conditioning information such as visual stimuli.

Recently, probabilistic generative models that leverage deep neural networks have made a significant impact across various domains, transforming the way we approach tasks such as image and audio generation[7–9] and time series imputation[10] as well as scientific applications ranging from molecular design[11] to black hole imaging.[12] Neuroscience research, in particular, has benefited from various types of deep generative models. For example, variational autoencoders (VAEs) have been applied to infer low-dimensional representations of single-trial neural population dynamics,[13] while generative adversarial networks (GANs) have been used for the task of spike-train generation[14,15] and EEG generation,[16–18] as well as to decode images from single neuron and fMRI data.[19,20]

Denoising diffusion probabilistic models (DDPMs)[21]—or diffusion models for short—have recently become the generative model of choice thanks to their superior performance compared to VAEs and GANs in many applications.[8,22,23] They have also been applied in neuroscientific settings, such as leveraging latent diffusion models[9] to predict viewed images from fMRI data in a neural decoding context.[24,25] However, diffusion models have been underexplored for the realistic generation of a type of data that is ubiquitous in neuroscience: multivariate and densely sampled electrophysiological time series. Such recordings include noninvasive scalp EEG as well as intracranial electrocorticography (ECoG) and local field potential (LFP), which are routinely recorded during scientific research and clinical brain monitoring.

A general-purpose model that can conditionally synthesize such neurophysiological data would be valuable in many scenarios. For example, brain recordings often contain missing values due to sensor noise or misplacement,[26] which limits their use in neuroscience-specific applications and can lead to them being discarded altogether. Such practical problems are exacerbated by the fact that individual neurophysiological datasets are often limited in availability due to restrictions in clinical contexts. Deep generative models for time series can partially alleviate these problems by providing synthetic data, for example, to facilitate the training of a brain-computer interface (BCI) or as input data for scientific simulations with reduced privacy concerns, or by imputing missing channels in high-dimensional recordings.[27–30] While diffusion models have been used to model complex high-dimensional time series in a variety of settings,[23] including probabilistic imputation[31,32] and forecasting,[33,34] and more specific applications in neuroscience, such as data augmentation for seizure classification,[35] their general utility and performance in modeling neurophysiological recordings have not been explored.

In this work, we demonstrate that diffusion models can accurately generate multivariate and densely sampled continuous neurophysiological recordings and further showcase their utility in a variety of neuroscience-specific applications. In particular, we use deep neural networks with structured convolutions,[36] a class of highly expressive sequence-to-sequence models,[37,38] and Ornstein-Uhlenbeck (OU) processes as diffusion processes[34] to improve the modeling of such time series with DDPMs. Our trained models are capable of producing synthetic recordings with realistic power spectra and even complex features such as sharp wave ripples (SWRs) and cross-frequency coupling. By conditioning on observed values and other behavioral or experimental variables, synthetic recordings automatically capture cross-area interactions between many channels, which are used to impute missing channels and improve performance in a human BCI application at test time. Finally, DDPMs provide additional benefits accessible to generative models, enabling behavioral state classification and outlier detection without additional network training. The code for our method and to reproduce our existing results is available at https://github.com/mackelab/neural_timeseries_diffusion.

RESULTS

DDPMs for neurophysiological time series

We use DDPMs[21] to learn generative models of electrophysiological recordings. DDPMs are defined by a forward (noising) process and a reverse (denoising) process. The noising process successively adds random noise from a prespecified distribution (usually independent Gaussian) to the real data through a series of diffusion steps. The reverse process uses a deep neural network as a "denoiser," which is trained to reverse the forward process at each diffusion step (Figure 1A). To generate synthetic data, samples of random Gaussian noise are drawn and gradually denoised by the trained network (Figure 1A, box, top to bottom) over the same number of diffusion steps, resulting in synthetic data that follow the statistical distribution of the real data (i.e., unconditional generation or simulation; Figure 1B). The training and inference procedure can also incorporate additional information through conditioning. For example, when real data from other channels are given alongside the initial noise sample for the generated channel, the denoising process is guided to generate synthetic data that are likely given the other channels, which can be used to perform imputation of missing channels or interpolation (Figure 1C). Similarly, when behavioral or experimental state variables are included during training, they can be used to guide the denoising process during time series generation (Figure 1D). In addition, such a conditionally trained model can also act as a classifier or outlier detector since it can be queried for the likelihood of observing a data sample given either condition.

We follow the standard DDPM training and inference procedure[21] while incorporating two recent innovations in time series modeling: first, since neurophysiological time series are densely sampled (e.g., sampling rate of 500 Hz or more), learning long-range dependencies over hundreds of time points in just 1 s of data is a nontrivial challenge. We address this problem by using structured convolutions in the denoising network, which use a collection of multiscale convolution kernels that have shown performance and efficiency improvements on long-range time series tasks.[36] Second, neurophysiological recordings of all modalities (EEG, ECoG, LFP) follow a $1/f$-like power law in the frequency domain under a range of behavioral and brain states,[39–42] in contrast to the flat spectrum of the Gaussian white noise commonly used in DDPMs. Therefore, we sometimes use the OU process (i.e., colored noise) in the forward noising

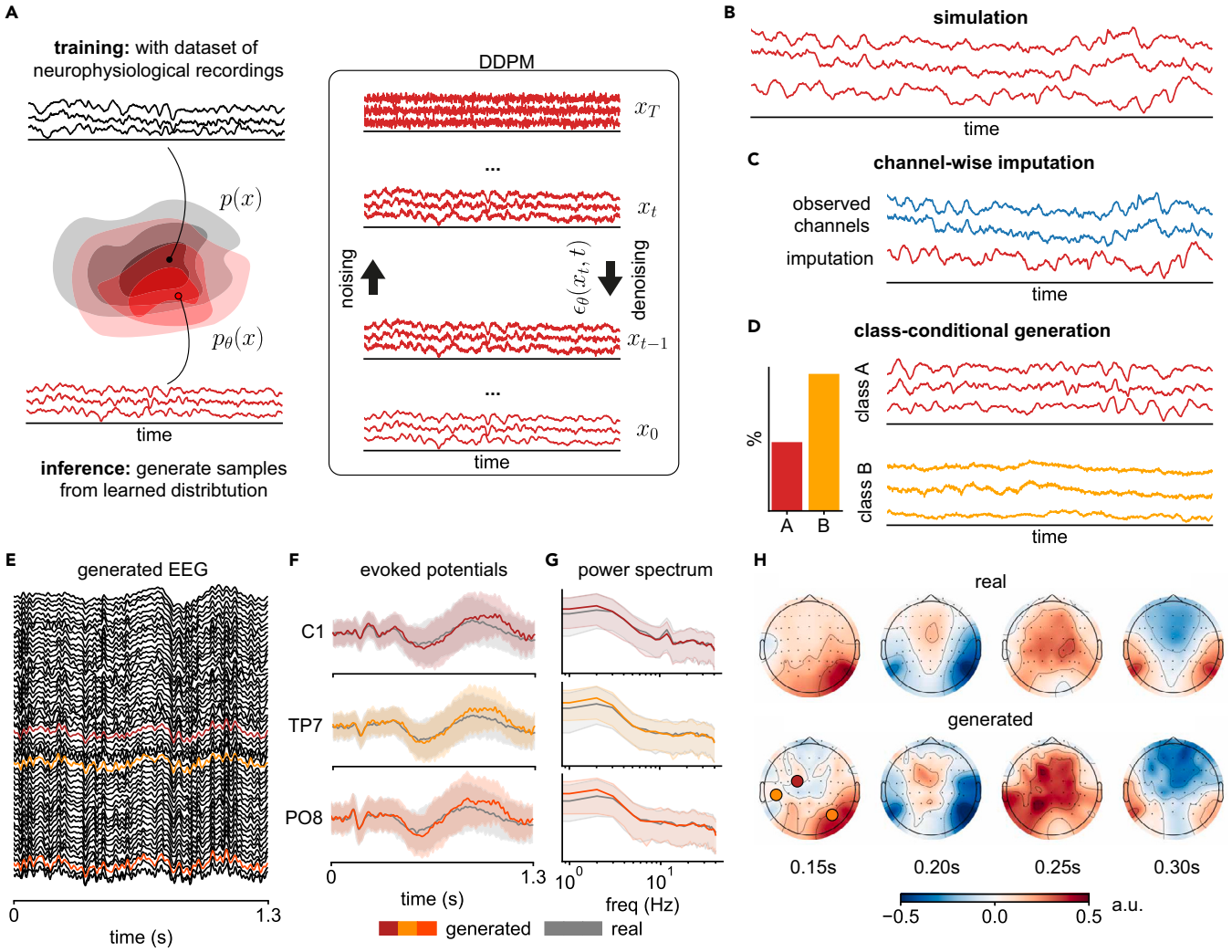


Figure 1. Overview of diffusion models for neurophysiological recordings and subsequent applications and an example of DDPM-generated EEG signal

(A) A denoising diffusion probabilistic model (DDPM) $p_\theta(x)$ is trained on a dataset of neurophysiological recordings. It attempts to generate samples from the data distribution $p(x)$, underlying the training data, by successively denoising samples from a prespecified Gaussian distribution, using a neural network $\epsilon_\theta(x_t, t)$ as the denoiser. In the context of neurophysiological recordings, DDPMs can be used for various different tasks.

(B–D) Examples include (B) simulation of neurophysiological recordings, (C) imputation of missing values in these recordings, and (D) class-conditional generation of recordings from different experimental conditions or brain states. Since DDPMs allow the computation of likelihoods, the class-conditional model can also be used to perform tasks like classification or outlier detection.

(E) An example DDPM-generated trial of 56-channel EEG.

(F–H) Trial average of three channels show close overlap between real (gray) and generated (colored) (F) evoked potentials (mean and standard deviation across trials), (G) power spectra (median and 10%/90% percentiles), and (H) spatiotemporal relationships reflected in scalp topography.

process[34] to incorporate this prior knowledge and empirically demonstrate improvements over the standard white noise. Full details on DDPMs, as well as our denoising network architecture and parameterization of the OU process, can be found in the experimental procedures.

As a first demonstration, we train a diffusion model on a single participant's EEG recorded during a trial-structured task (BCI Challenge @ NER 2015, Margaux et al.[43]). The trained DDPM is able to simulate (i.e., unconditionally generate) realistic single-trial EEG data (Figure 1E) and capture features in the real trial-averaged time series and power spectra (Figures 1F and 1G), in particular the evolution of the evoked potentials and a 12 Hz oscillation in the C1 channel. In addition, spatiotemporal relationships across the entire scalp are preserved, as shown in the topographic plots over time (Figure 1H). This introductory demonstration illustrates how DDPMs can realistically generate high-dimensional neurophysiological time series, and in the following sections, we further evaluate and exploit this capability on a variety of datasets and applications.

Overview of DDPM applications: Experiments and datasets

Beyond the first example, we apply our model to three different datasets to test whether diffusion models can generate realistic

synthetic brain recordings in different settings and demonstrate their utility in a variety of tasks relevant to neuroscience. On all three datasets, DDPMs accurately capture the statistics of real recordings—in particular, the distribution of channel-wise power spectral densities (PSDs), as well as the multivariate cross-spectrum—across species, recording modalities, and brain areas. Critically, the trained models do not simply overfit to or memorize samples from the training set, which we confirm by measuring the pairwise distance between generated and training data in the time and frequency domains (Figure S4). We show that DDPMs are a general-purpose generative model that accurately capture idiosyncratic features in each of the three datasets and can be applied in a variety of tasks with minor modifications to network architecture or training procedure.

The first dataset consists of 3-channel LFP recordings from rat prefrontal cortex, thalamus, and hippocampus.[44] These recordings contain SWRs that exhibit complex cross-frequency and cross-channel dependencies, which are correctly captured by our model in both simulation and imputation settings. The second dataset consists of human ECoG recorded during naturalistic behavior from 12 participants undergoing epilepsy monitoring (AJILE12, Peterson et al.[45]). In addition to accurately generating high-channel-count ECoG data, we show that DDPMs can sensibly impute corrupted or missing channels in an artificially induced missing-data setting (similar to Talukder et al.[30]), which substantially improves performance in a neural decoding task compared to a baseline (mean) imputation. The third dataset consists of whole-brain ECoG recordings from a macaque monkey under two different brain states (awake and anesthetized). DDPMs can conditionally generate recordings given one of these two states, as well as evaluate class-specific likelihoods, which in turn can be used to perform classification and outlier detection. The rest of this section presents detailed results from each of the three experiments.

DDPM captures cross-frequency and cross-region dependencies of SWRs

In the first experiment, we apply our DDPM to model LFPs recorded from freely behaving rats (sampled at 600 Hz) in three locations: medial prefrontal cortex (mPFC), hippocampal CA1 region, and the thalamic nucleus reuniens (RE).[46] These recordings contain neurophysiological features of interest, such as slow oscillations, spindles, and SWRs, which are region specific and occur with a specific phase relationship relative to one another.[44] Thus, we aim to generate synthetic data and assess whether our model is able to correctly capture the single-channel voltage distribution and power spectrum as well as complex features such as phase-amplitude coupling (PAC) between slow oscillations and ripples across channels. In addition, to compare the fidelity of generated recordings using different types of generative models, we also train a VAE[47] and a GAN[48] from a recently published study on generative modeling of electrophysiological brain recordings.[16] Furthermore, we improve both models by incorporating more competitive encoder/decoder and generator/discriminator architectures based on the structured convolutions used in our denoiser architecture.

After training models on 3,480 2-s time series, we generate the same number of synthetic samples for comparison. Visually, DDPM-generated traces look indistinguishable from real data and contain clearly visible SWRs (example in Figure 2B). Across all generated samples, the distribution of voltage values and PSDs between real and synthetic data match closely in all three regions (Figure 2C), and components such as 60 Hz line noise are also reproduced correctly. For the VAE and GAN baselines, only the models that use our architectures based on structured convolutions are capable of producing visually consistent samples (see Figure S1). Nevertheless, VAE samples are overly smooth (i.e., low-frequency dominated), while GAN samples contain frequency characteristics not found in the real data and fail to capture finer details (Figures S2 and S3).

To assess whether our DDPM can reproduce more complex cross-regional and ripple-specific features in the real data, we measure the degree of PAC between the amplitude of the CA1 ripple band (100–275 Hz) and the phase of delta (1–4 Hz), theta (4–9 Hz), and spindle (6–14 Hz) frequencies in the mPFC. We additionally isolate this analysis to the actual occurrences of ripples by thresholding the ripple band power and computing a "phase count coupling" histogram between detected CA1 ripples and the instantaneous phase of the aforementioned driver frequencies in the mPFC (similar to Varela and Wilson[44]). Overall, we see that ripples in both generated and real data have remarkably similar phase preferences and, for all three frequencies in the mPFC, peak between $-45°$ and $45°$ (Figure 2D). As a baseline comparison, we also created channel-wise surrogates[5] that preserve the single-channel PSD but do not show any preferred phase coupling. Similarly, the recordings generated by the VAE and GAN do not exhibit the correct PAC (Figures S2 and S3).

While the above demonstrates that DDPM-generated samples reproduce the correct temporal relationships between CA1 ripples and slower fluctuations in the mPFC, we further investigate whether SWRs are correctly captured by our model by conditionally generating (i.e., imputing) CA1 traces given real mPFC and RE traces for a set of held-out evaluation time series. In other words, when provided with the other channels as context, can our model correctly infer when a ripple would have occurred in CA1? We use a technique known as "inpainting" to impute the CA1 channel.[49] This technique allows DDPMs to fill in missing values by guiding the diffusion process along observed values in the other dimensions (i.e., channels). To evaluate performance for a given (held-out) evaluation sample, we check if both the real and imputed CA1 traces contain a ripple by thresholding the ripple band power and then compute the F1 score for this classification task across the whole evaluation set. To test whether our model is better than chance at generating CA1 ripples given the appropriate context, we perform a permutation test by shuffling the binary labels of the real CA1 data (i.e., "was there a ripple or not?") and recomputing the F1 score.[50]

We provide an example of a successfully imputed ripple (Figure 2E), where the generated CA1 trace contains a ripple at exactly the same time as when the real ripple occurred around the 800 ms mark. Repeating the imputation procedure 100 times on the same mPFC and RE traces, we see that the average ripple band power in the imputed CA1 traces is highest at the time that coincides with when the real ripple occurred (Figure 2F). Across all evaluation samples, our model conditionally generates ripples

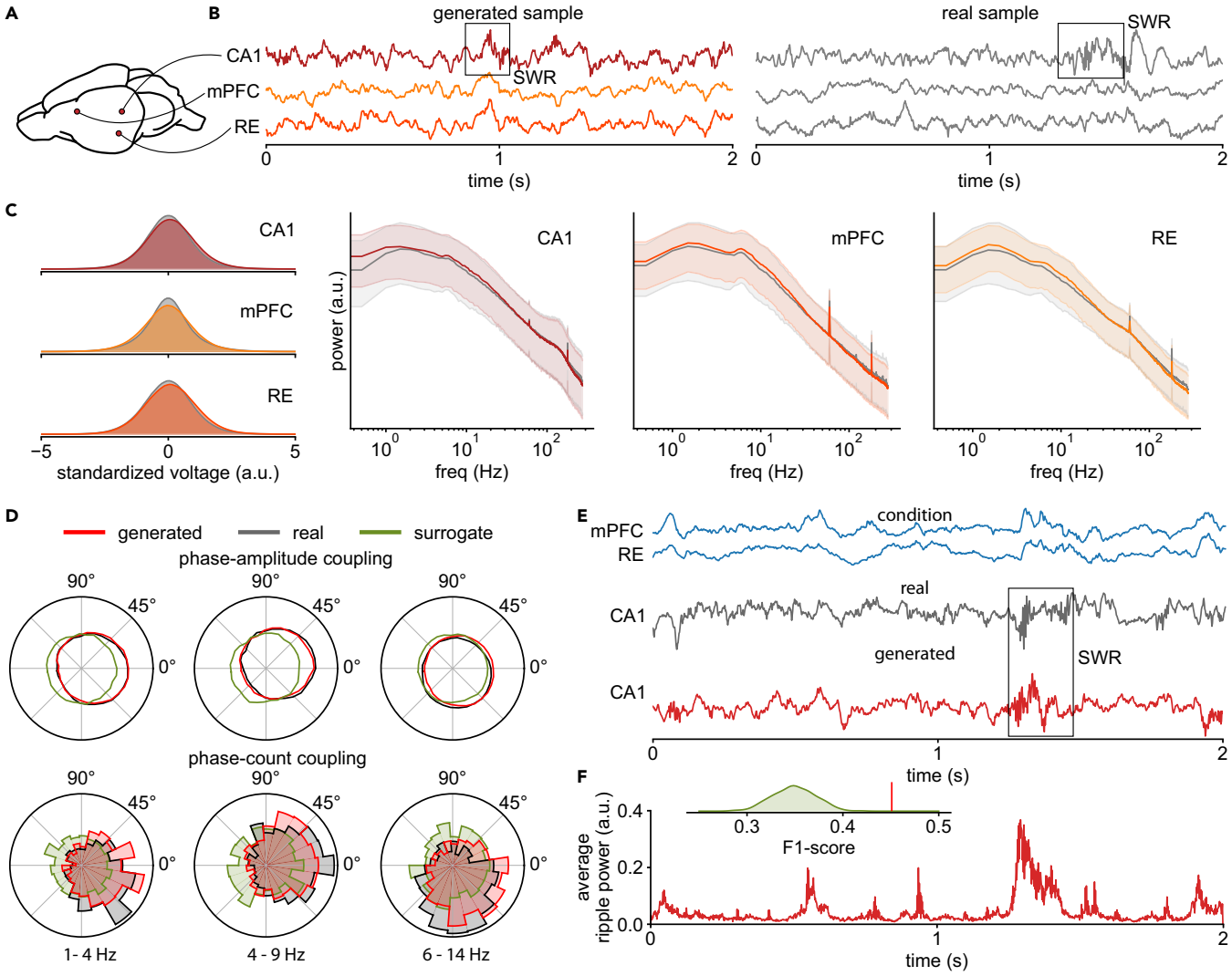


Figure 2. DDPM captures rat cortical-hippocampal LFP dynamics

(A) Schematic of rat brain showing recording locations: CA1, mPFC, and RE.

(B) Example of time series generated by our model (left) and a real sample (right). Both examples contain SWRs.

(C) Marginal distributions over the standardized voltage of each channel for both real and generated data (left), as well as median and 10%/90% percentiles of real and generated power spectra for each channel (right).

(D) Phase-amplitude and phase-count coupling of CA1 ripples for different driver frequencies in the mPFC for real, surrogate, and DDPM-generated data.

(E) Example of conditionally generated CA1 trace given real mPFC and RE traces. The real and conditionally generated traces contain a ripple in the same position.

(F) Average power in the SWR frequency band (100–275 Hz) over 100 samples from the model (conditioned on the same mPFC and RE traces as in E) has a large peak at the same position, indicating that many SWRs are generated at this location. Inset: across the evaluation dataset, the prediction performance ("is an SWR present in both the real and predicted time series?") measured in terms of F1 score is significantly different from the distribution of scores for randomly permuted predictions.

significantly better than chance (F1 = 0.45, $p_{perm} < 0.001$; Figure 2F, inset). Thus, our model successfully captures within-channel temporal characteristics, as well as cross-frequency and cross-channel relationships observed in rat LFPs during SWR occurrences, and can therefore be used for probabilistic generation or simulation of highly complex brain recordings. This ability to produce high-quality simulations also extends to more high-dimensional examples such as the 56-channel EEG data (Figures 1E–1H) or the 128-channel macaque ECoG data (Figure S6).

DDPM-generated imputations improve neural decoding with missing data

In our second experiment, we use the AJILE12 dataset,[45] which consists of ECoG recordings from twelve human participants during naturalistic behavior while undergoing epilepsy monitoring (electrode coverage of one example participant is shown in Figure 3A). Using an artificially induced missing-data scenario (following a similar setup as in Talukder et al.[30]), where only a subset of the recorded channels are observed and the goal is to predict the missing channels from the observed ones, we

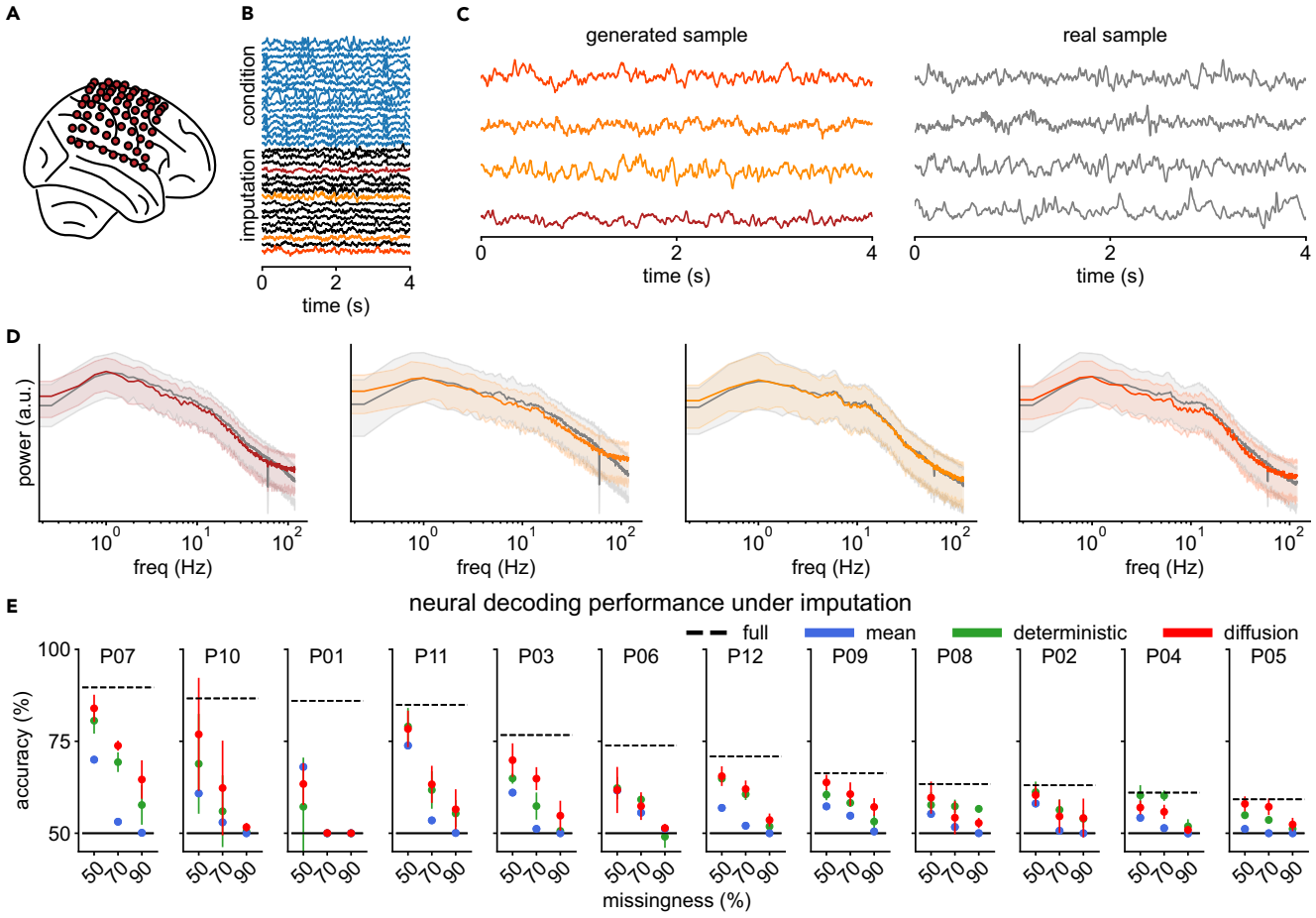


Figure 3. DDPM-generated imputations of human ECoG recordings improve BCI decoding

(A) Electrode layout of participant P07 from the AJILE12 dataset.

(B) Example imputation of a 4-s window from P07. The first 32 channels are used as conditioning information (blue), while the remaining 32 channels are imputed (conditionally generated); four are highlighted and shown in (C).

(C) Four randomly selected channels from the imputation (left) together with the corresponding real sample (right).

(D) Median and 10%/90% percentiles of real and generated power spectra for the four channels.

(E) Test performance of the neural decoding model for each of the 12 participants with fully observed data (dashed) as well mean-imputation (blue), deterministic-neural-network-based imputation (green), and DDPM-based imputation (red) under different amounts of missingness. Participants are sorted by the decoder performance on fully observed data. Mean and standard deviation over five different randomly drawn missing channel configurations are shown.

demonstrate how diffusion-based imputation can subsequently improve classifier performance in a neural decoding task.

For each participant, we train a diffusion model on all channels from their first 6 days of recording, with the number of 4-s-long training time windows varying between 148 and 1,512 per participant. Unlike the model used for rat LFP data, the diffusion model for each AJILE participant is trained to directly capture the conditional distribution of missing channels given observed channels. This is achieved by randomly dropping out channels during training and conditioning on the remaining channels to perform imputation. Conditional training can further improve imputation performance over the previously used unconditional training.[31] Details on the dataset, network architecture, and conditional versus unconditional training are given in the experimental procedures.

After training the model, we test its ability to impute "missing" channels of held-out evaluation samples from the seventh day of recording by generating synthetic data for half the channels while conditioning on real data from the other half (i.e., 50% missingness; Figure 3B). Conditionally generated synthetic traces look visually similar to real data in the corresponding channels (Figure 3C, 4 out of 64 channels highlighted), while synthetic PSDs closely match those of the real data for the imputed channels across all evaluation time series (Figures 3D, S8, and S9), demonstrating that DDPMs can be used for participant-specific imputation even with a substantial amount of channels missing. For the AJILE12 dataset, the quality of imputed traces is improved by using OU processes instead of white noise (experiments on the effect of using the OU process versus white noise are described in the supplemental information; Figures S10–S12).

To further quantify the quality of the imputations, we compute the correlations between the ground truth and imputed channels (as in Talukder et al.[30]). For each participant, 50%, 70%, and 90% of the channels are randomly dropped from all time series of the evaluation set and then probabilistically imputed using

the trained diffusion model (as in Figure 3B). This procedure is repeated with five different random seeds for each dropout rate. We also compare the diffusion-based imputation to a mean-imputation baseline—a strategy often employed in such neural decoding paradigms when faced with missing data.[30] Furthermore, to provide a stronger baseline, we perform neural-network-based imputation, where we use a regression network based on our denoiser architecture with structured convolutions and train it to perform deterministic imputation by minimizing the mean-squared error (an example of deterministic imputation is shown in Figure S5).

For a dropout rate of 50%, the average channel-wise correlation across all participants and test signals is 0.27 for DDPM-generated samples. The imputation performance varies widely between participants, with the best one achieving an average correlation of 0.42 and the worst a correlation of 0.11. As expected, with increasing dropout rates (70%, 90%), the correlation further drops. In comparison, mean imputations have a correlation of zero by definition. With an average correlation of 0.50 for a dropout rate of 50%, the deterministic, neural-network-based imputations are more correlated to the true data than the probabilistic DDDP-based imputations (see numerical comparison in Table S1). However, as shown next, this does not necessarily translate into higher performance in neural decoding tasks.

To demonstrate the utility of diffusion-based imputation, we use a binary neural decoding task of classifying the 4-s-long time series that were recorded under rest from those recorded during movement. We use per-participant random forest classifiers that were also trained on the first 6 days of each participant's recording as decoders (similar to Peterson et al.[51]) and evaluate classification performance on fully observed evaluation time series to establish a performance upper bound (Figure 3E, dashed). We then classify the imputed time series using the random forest classifier and report the average and standard deviation across dropout random seeds (Figure 3E, red).

Across 12 participants and 3 missingness levels, we observe that diffusion-based imputation (Figure 3E, red) leads to better classification performance than mean imputation (Figure 3E, blue) in all participants, apart from P01 and P06. In addition, the DDPM-based, probabilistic imputations achieve a decoding accuracy that is competitive with the neural-network-based, deterministic imputations (Figure 3E, green): the increase in decoding accuracy over the mean imputations is larger for the DDPM-based imputations 8 out of 12 times for both 50% and 70% dropout rates and 9 out of 12 times for a 90% dropout rate (see a numerical comparison for all participants and dropout levels in Table S1).

Similarly, the DDPM compares favorably to the decoding performance of the deterministic imputations from the autoencoder-based models from Talukder et al.,[30] which are specifically tailored to perform imputation (numerical comparison in Table S1). Overall, like the channel-wise correlations between ground truth and imputed time series, the decoding performances after imputation appear highly participant dependent. Nevertheless, diffusion-based imputation significantly improves neural decoding accuracy, even with a significant proportion of channels missing.

Class-conditional DDPMs can generate and evaluate brain-state-dependent recordings

In our third experiment, we use the DDPM to model whole-brain surface ECoG from a nonhuman primate (macaque monkey) sampled at 1,000 Hz.[52] In this experiment, we model 12 randomly selected channels (out of 128; Figure 4A) during two distinct behavioral states: awake, a control condition where the animal is head and arm restrained but otherwise freely behaving, and anesthetized, where the animal is fully unconscious following the induction of general anesthesia. Our class-conditional diffusion model can accurately generate synthetic recordings under both states, which can be used to train a classifier without giving it access to real data. In addition, we can evaluate likelihoods under the learned generative model, which can be used for state classification and outlier detection of real recordings.

After training our model on a total of 870 1-s-long windows from both the awake and anesthetized conditions, we conditionally generate the same number of synthetic time series for evaluation. For both the awake and anesthetized conditions, the real and generated example time series are visually indistinguishable from each other (Figure 4B, 4 out of 12 channels are highlighted). Notably, synthetic data during the awake state contain more higher-frequency fluctuations and oscillations lasting several cycles, while abrupt lower-frequency fluctuations dominate during the anesthetized state. Across all 12 modeled channels, the PSDs of real and synthetic data closely match in both conditions (Figure 4C).

In addition to generating synthetic recordings in a state-dependent manner, class-conditional DDPMs can also evaluate the likelihood of recordings under each condition; that is, how likely is it to observe a given recording under awake versus anesthetized states? Using the difference of log likelihoods from our trained models to classify previously unseen evaluation time windows from either state, we achieve an evaluation accuracy of 83% on the balanced classification task. Additionally, when we evaluate likelihoods over a 1.5-h segment of the recording in a sliding-window manner, we observe that the difference in log likelihood between anesthetized and awake conditions is negative to start and then sharply increases after the induction of general anesthesia—even though the diffusion model does not have access to this information, nor was it trained on nearby segments of the recording (Figure 4D, training segments in blue). As time elapses, the difference of log likelihoods slowly decreases toward the "awake level" (negative) as the anesthesia wears off, demonstrating how such generative models can be applied for continuous state prediction.

Additionally, unlike with purely discriminative classifiers, the ability to evaluate likelihoods allows us to use our generative model to perform outlier detection, which we demonstrate through two experiments. In the first experiment, for all time series in the evaluation set, we replace half of the channels with white noise to simulate complete signal loss in some of the recording electrodes. In the second experiment, we instead reverse half of the channels time-wise, which is unlikely to happen in the real world but creates time series that are visually and statistically similar to the real data, thus representing a difficult out-of-distribution test scenario. Using the likelihood of the real and outlier data provided by the diffusion model as a score for classification, the area under the receiver operator

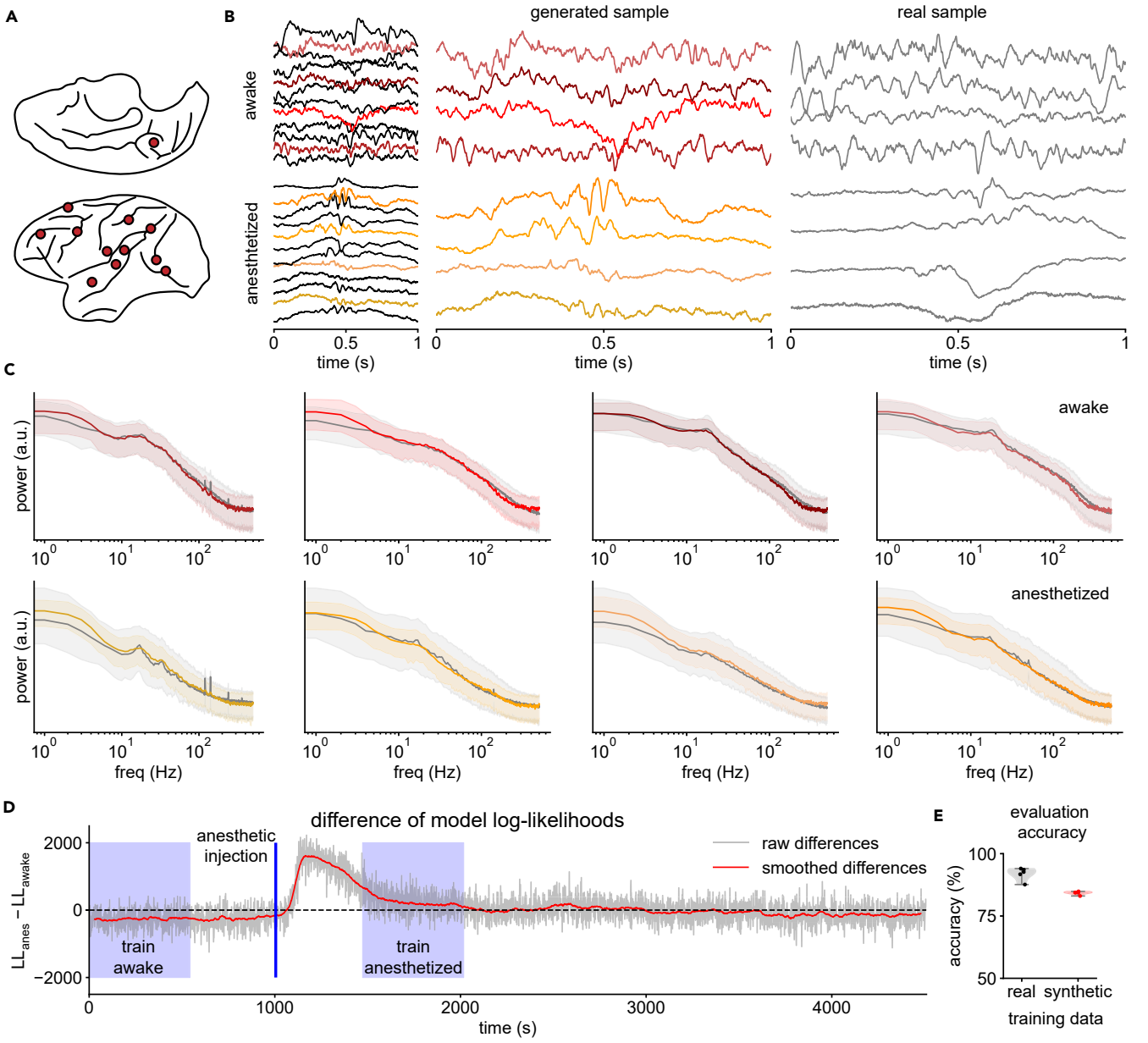


Figure 4. Conditional DDPM learns brain state-dependent macaque ECoG recordings

(A) Electrode locations in the macaque brain of the 12 randomly selected channels used for modeling.

(B) Examples of conditionally generated time series for both the awake and anesthetized conditions. Four randomly selected channels are shown in more detail (left) together with real samples (right).

(C) Median and 10%/90% percentiles of real and generated PSDs for the four channels stratified by condition.

(D) Difference of log likelihoods over 1.5 h of recording. A positive difference indicates a classification toward "anesthetized" and a negative difference toward "awake." Periods used for model training are shaded in blue.

(E) Real data test accuracy of a brain-state classifier trained on real (black) versus DDPM-generated synthetic data (red).

characteristic (AUC) for time-wise-flipped outliers is 0.63, while the AUC for white noise outliers is 1.0. When comparing pairs of time-wise flipped outliers with their corresponding real time series, the likelihood of the real time series is larger in 98% of the cases, demonstrating the model's ability to distinguish data that have the same marginal distribution per channel but are nevertheless out of distribution.

Finally, to further demonstrate the utility of using diffusion models to generate realistic neurophysiological recordings, we perform brain-state classification based on synthetic data. In many situations, the original data are sensitive due to privacy concerns, for example, when collected as a part of medical examination, and therefore cannot be publicly shared but may otherwise be useful for applications such as training of biometric algorithms. In such situations, synthetic recordings that preserve specific aspects of the recording while anonymizing others may be valuable. Here, we use the trained DDPM to generate synthetic data from awake and anesthetized conditions and train a

neural-network-based discriminative classifier (as described in Wang et al.[53]) solely on these synthetic data. The trained classifier is then evaluated on a held-out evaluation set of real recordings while never having seen real data before. Since the diffusion model captures the conditional distribution of the two classes well, the classifier trained on synthetic data achieves an accuracy of 84% when classifying real data (Figure 4E), which is close to the accuracy of a classifier trained on real data (92%). We thus demonstrate the ability to create artificial datasets that can be shared to facilitate further research with reduced privacy concerns.

DISCUSSION

Summary

Our experiments show that DDPMs, together with a flexible convolutional denoising network and the use of the OU processes, are a powerful tool that can accurately model highly multivariate and densely sampled neurophysiological data, providing features and, more importantly, realistic synthetic data useful for applications relevant to neuroscience. The power of DDPMs lies in their generality, as they provide a general-purpose generative model that can be used for many different applications ranging from unconditional (i.e., simulation) and conditional generation to imputation and likelihood evaluation.

We demonstrate these applications here on three different datasets of neurophysiological recordings with channel counts ranging from 3 to 128 channels from a variety of brain regions, recording modalities, and species. Overall, our models accurately capture data distributions in the time and frequency domains, as well as detailed cross-channel and cross-frequency dependencies like SWRs. On an imputation task, they achieve a performance that is on par with or better than tailored imputation methods for neurophysiological recordings. Finally, we show successful applications of class-conditional models in classification, outlier detection, and synthetic training data generation.

These features make DDPMs a viable tool for neuroscientists and clinicians studying the complex dynamics embedded in neurophysiological recordings. After successful training, DDPMs can be used to generate unlimited amounts of data as input to neuroscience simulators. When neurophysiological data are subject to strict privacy requirements, as may be the case with human clinical data, neuroscientists and clinicians can use DDPMs to generate artificial datasets that can be shared with collaborators with less concern. In addition, the accurate imputations provided by DDPMs can eliminate the need to discard large amounts of data or modify an already trained neural decoder. Finally, DDPMs output log likelihoods that provide users with a model to perform simple classification tasks or outlier detection at no additional training cost.

Other potential applications

Trained DDPMs can be used for a variety of other applications that we do not show in this work, such as training data augmentation, as well as imputation of more complex missingness patterns, which includes the task of time series forecasting.

Instead of completely replacing training datasets with artificial ones, as described in the data privacy use case (Figure 4E), it is possible to augment existing real training data with synthetic data generated by a DDPM to improve the performance of another model (e.g., a classifier). This additional training data could lead to performance improvements, especially if the relevant model is difficult to regularize and prone to overfitting. DDPMs can provide tailored noise-augmented data samples that capture complex relationships other surrogate methods may not. Throughout our experiments, DDPMs were robust to overfitting and able to achieve good sample quality with only a few hundred training time windows of 1- to 4-s long. In addition, DDPMs can be trained on a separate, larger dataset to generate synthetic "pretraining" data for a classifier model that is later fine-tuned on task-specific but smaller datasets, that is, to facilitate transfer learning. Finally, DDPMs might be used to conditionally generate data with specific properties to serve as null models or realistic control data for neuroscientific measures, such as those measuring neural coupling in hyperscanning experiments.[54,55]

DDPMs can also handle more complex missingness patterns. In this work, we have demonstrated the ability of DDPMs to perform cross-channel imputation, although completely missing channels represent only one pattern of missingness that can occur in real neurophysiological recordings (i.e., dead electrode). Many other patterns of missingness are possible, such as when some time points within a recorded channel are missing due to momentary artifacts. Imputation in this context would mean filling in these missing measurements given information from other channels and observed values in the channel itself. These more complex missingness patterns can be realized with diffusion models under both training strategies: for unconditional training, all possible missingness patterns work out of the box via inpainting (i.e., Figure 2E). For conditional training, the training strategy needs to be generalized from dropping out whole channels to more complex missingness patterns. Tashiro et al.[31] discuss several strategies for this purpose. A special case is the common task of time series forecasting, where the missing values are simply unobserved time points in the future. Like imputation, time series forecasting can then be realized by diffusion models trained unconditionally or conditionally.

An additional benefit of our fully convolutional denoising network is that it allows users to generate neurophysiological recordings of arbitrary length despite training on fixed-size time series. To generate long synthetic recordings, the denoising network is simply applied to the entire time series without any further technical modifications (though dependencies longer than the timescale of the training segment may not be correctly modeled). We have not used this capability here, but it will be important when generating input to a simulation where long synthetic recordings are required.

Limitations

Although diffusion models are a very popular type of deep generative models today, they are not without limitations. Because DDPMs generate samples by successively denoising them over (typically) hundreds of denoising steps, inference is computationally intensive, especially when compared to other families of deep generative models such as VAEs or GANs. Much recent work has focused on speeding up the inference time of diffusion models, often by tolerating a small degradation in sample quality.[56–58] While in this work, we applied the standard formulation

of DDPMs without further speedup, all progress in this direction is directly applicable to our setting and provides a potential remedy for the comparatively long inference times.

Furthermore, diffusion models typically operate on the dimensionality of the original data space. In neuroscience, however, there has been much interest in lower-dimensional state-space representations of high-dimensional neurophysiological recordings, especially neuronal population spiking data,[59] but more recently also continuous time series such as LFPs.[60] One way to obtain such low-dimensional neural representations with generative models is to use VAEs (e.g., like in Pandarinath et al.[13]), which model the original data distribution via a mapping to a lower-dimensional space of fixed dimensionality. This encoding property is not automatically fulfilled by DDPMs, but combinations of encoder-decoder architectures with a diffusion model in the latent space have been successfully applied to image generation and fMRI decoding.[9,25] Similar approaches are possible for neurophysiological recordings and may provide low-dimensional representations, along with increased computational efficiency. In particular, a shared latent space could be beneficial in applications similar to our imputation experiment on the AJILE12 dataset (Figure 3). Unlike the autoencoder-based model described in Talukder et al.,[30] we trained an individual DDPM for each of the twelve participants. Therefore, a potential extension of our current model is to jointly train a latent diffusion model using participant-specific encoder and decoder layers, which could increase performance and robustness due to its ability to generalize over different participants. In addition, imputation performance for the AJILE12 dataset might be improved further with more targeted variations of our architecture following the architectural principles proposed in, for example, Tashiro et al.[31] or Alcaraz and Strodthoff[32] or by applying recent diffusion-based methods to solve inverse problems.[61]

As discussed earlier, synthetic training data can alleviate privacy concerns. However, our DDPM-generated synthetic data are private in the sense that the minimum difference between time series in the real training data is similar to the minimum difference between time series in the real and generated data. While intuitive, this measure does not provide theoretical guarantees against any form of reverse engineering or memorization.[62]

Lastly, a persisting challenge in modeling neurophysiological recordings is to capture very long-range dependencies over timescales of minutes or even hours. In our training setup, the time series extracted from a long neurophysiological recording are treated as independent samples by the model. Thus, temporal dependencies beyond the length of the time series used for training are not captured. Naturally, for any given dataset, long-range and cross-scale dependencies are much harder to capture due to data sparsity. Thus, building deep generative models that are more data efficient and can capture long-range dependencies by training on only a few samples is a challenging but exciting avenue for further research.

Conclusion

Together with a flexible convolutional denoising network and the use of the OU processes, DDPMs provide a powerful and flexible type of generative model for neurophysiological recordings that can generate high-quality samples and be used in a variety of applications relevant to neuroscience like simulation, imputation, or brain-state classification.

EXPERIMENTAL PROCEDURES

Resource availability

Lead contact

Further information and requests for resources should be directed to and will be fulfilled by the lead contact Julius Vetter (julius.vetter@uni-tuebingen.de).

Materials availability

This study did not generate any new materials.

Data and code availability

- Code for our method and to reproduce our existing results is available at https://github.com/mackelab/neural_timeseries_diffusion and has been deposited at Zenodo.[63]
- All four datasets used in this work are publicly available: rat LFP,[46] https://crcns.org/data-sets/hc/hc-24; AJILE12,[45] https://dandiarchive.org/dandiset/000055/0.220127.0436; macaque ECoG,[52] http://www.neurotycho.org/anesthesia-task; and BCI Challenge @ NER 2015,[43] https://www.kaggle.com/competitions/inria-bci-challenge/.

DDPMs

To train our diffusion models, we consider a training dataset of n time windows extracted from neurophysiological recordings $\{\mathbf{x}^{(i)}\}_{i=1}^{n}$, where $\mathbf{x}^{(i)} \in \mathbb{R}^{C\times L}$ with C channels of fixed length L time steps. During sampling or inference, arbitrary time series lengths are admissible due to our fully convolutional network architecture.

Background on DDPMs

Our primary goal is to learn a parametrized probability distribution $p_\theta(\mathbf{x})$ on $\mathbb{R}^{C\times L}$ that is close to the original data distribution $p(\mathbf{x})$. $p_\theta(\mathbf{x})$ can then be used for many different tasks like imputation, generation, or likelihood evaluation. We use DDPMs.[21]

DDPMs work by modeling two processes, the reverse (denoising) and forward (noising) diffusion processes, over a sequence of latent variables $\mathbf{x}_t$, $t = 1,\ldots,T$, where T is a prespecified process length. Note that T is the number of diffusion steps, a hyperparameter in the generative model, and not the length of the time series being modeled (L above). The forward process is given by a Markov chain

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) := \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}), \qquad \text{(Equation 1)}$$

where the noising distribution

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) := \mathcal{N}\left(\mathbf{x}_t; \sqrt{1-\beta_t}\,\mathbf{x}_{t-1}, \beta_t\mathbf{I}\right) \qquad \text{(Equation 2)}$$

iteratively adds Gaussian noise controlled by the noise level $\beta_t > 0$, resulting in an evolution from the data distribution, $p(\mathbf{x}_0)$, to the noise distribution, $p(\mathbf{x}_T)$, over the course of T diffusion steps. This forward process is chosen by the user beforehand and thus not learned.

As its name suggests, the reverse process reverses the forward process and is given by

$$p_\theta(\mathbf{x}_{0:T}) := p(\mathbf{x}_T)\prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t), \mathbf{x}_T \sim \mathcal{N}(0,\mathbf{I}), \qquad \text{(Equation 3)}$$

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t,t), \sigma_\theta(\mathbf{x}_t,t)\mathbf{I}),$$

that is, the reverse conditional distribution is also Gaussian, and its mean and variance are parametrized functions of the current diffusion step t and the noised data point $\mathbf{x}_t$ and, therefore, need to be learned. DDPMs parameterize $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ as a denoising distribution,

$$\mu_\theta(\mathbf{x}_t,t) = \frac{1}{\alpha_t}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathbf{x}_t,t)\right), \qquad \text{(Equation 4)}$$

$$\sigma_\theta(\mathbf{x}_t, t) = \left(\frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\sqrt{\beta_t}\right)^{\frac{1}{2}},$$

where ϵ_θ is a neural network with learnable weights θ.

In other words, the neural network $\epsilon_\theta(\mathbf{x}_t, t, \text{cond})$ acts as denoiser that, together with the diffusion time step t, and, optionally, additional conditioning information cond, produces a denoised version of the data point (in this case a time series) as the basis for the next denoising step.

The denoising network can be optimized by computing a variational lower bound on $p_\theta(\mathbf{x}_t|\mathbf{x}_{t-1})$, which results in the full (simplified) training objective:

$$\min_\theta \mathcal{L}(\theta) := \min \mathbb{E}_{\mathbf{x}_0,\epsilon}\left[\| \epsilon - \epsilon_\theta(\mathbf{x}_t, t) \|_2^2\right], \quad \text{(Equation 5)}$$

$$\text{with } \mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + (1-\bar{\alpha}_t)\epsilon.$$

DDPMs for neurophysiological time series

For our application of modeling neurophysiological time series, we extend the standard DDPM setting with two modifications.

(1) First, we use a flexible convolutional network architecture based on structured convolutions[36] as our denoising network. Structured convolutions belong to a family of recent, highly expressive sequence-to-sequence models.[32,37,38] Our architecture is able to accurately model long (thousands of time points) and highly multivariate time series (over 100 channels).

(2) We sometimes replace the white noise process used in the standard DDPM with general Gaussian processes. This allows us to include desirable spectral properties into the generative process.[34] In this paper, we specifically focus on the OU process due to its advantageous numerical properties and relevance for neurophysiological recordings, particularly the inverse power law scaling over frequencies.

Network architecture with structured convolutions

Our denoising network consists of interleaved layers of structured convolutions[36] and linear layers that "mix" information between channels.[64]

Structured long convolutions are created by concatenating several linearly interpolated and exponentially downscaled convolutional kernels. The resulting convolutional kernel can be as long as the time series itself. This construction is inspired by recent advances in deep state-space models[32,37,38] and allows for long yet parameter-efficient convolutional kernels that can model complex and long-range dependencies within a given time series. To make the use of these large convolutional kernels tractable, the convolution operation is computed by element-wise multiplication in the Fourier domain after applying a fast Fourier transform. This approach drastically reduces the computation time for large kernels and long time series.

The input layer is a standard convolutional layer[65] that maps the input into a high-dimensional latent space. We parameterize the size of the latent space per channel; that is, we specify the number of latent dimensions per channel. The full dimensionality of the latent space is then given by the number of channels times the latent dimension per channel. The output layer is also a standard convolutional layer that maps the hidden activations to the desired output dimensionality. The hidden layers consist of blocks of structured convolutions followed by linear layers. The linear layers mix the output from the structured convolutions, which operate separately on each hidden channel. Unlike the original implementation of structured convolutions, we introduce the number of scales as a new hyperparameter. This means that the overall size of the structured convolutional kernel is given by $\text{kernel\_size} \cdot 2^{\text{num\_scales}-1}$.

In our experiments, especially when training on neurophysiological recordings with a high channel count, we noticed that fully parametrized linear mixing layers can cause convergence issues. The number of weights in the linear layers grows quadratically with the full latent dimensions and can quickly dwarf the number of weights in convolutional layers. To avoid this overparameterization and alleviate the convergence issues, we sparsify the weight matrices of the linear layers. To do so, we decrease the number of active weights in off-diagonal blocks, where the size of the off-diagonal blocks is a hyperparameter.

Throughout our architecture, we use Gaussian error linear unit (GELU) activation functions[66] and batch or layer normalization.[64,67,68] Importantly, our architecture does not perform any compression in time. This allows the denoising architecture to operate on time series of arbitrary length.

Alternative noise processes as diffusion process

It is straightforward to replace the white noise process commonly used in DDPMs with other Gaussian processes.[34] Instead of using white noise from $\mathcal{N}(0, I)$, the (forward) diffusion process is performed with noise sampled from a general Gaussian process $\mathcal{N}(0, \Sigma)$ with prespecified covariance Σ. The objective from Equation 5 is also changed to include Σ:

$$\min_\theta \mathcal{L}(\theta) := \min \mathbb{E}_{\mathbf{x}_0,\epsilon}\left[(\epsilon - \epsilon_\theta(\mathbf{x}_t, t))^T \Sigma^{-1} (\epsilon - \epsilon_\theta(\mathbf{x}_t, t))\right], \quad \text{(Equation 6)}$$

where $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + (1-\bar{\alpha}_t)\epsilon$ and $\epsilon \sim \mathcal{N}(0, \Sigma)$. This replacement is useful for providing the model with domain knowledge about the recordings and can lead to better empirical performance.

One such example of domain knowledge in the context of neurophysiological recordings is continuity. Neurophysiological recordings are typically continuous and do not exhibit large and abrupt jumps. In addition, for many neurophysiological recordings, the frequency spectrum exhibits inverse power-law scaling.[39]

In this paper, we use the OU process, which is a continuous noise process with a power-law frequency spectrum, to encode this domain knowledge.

Using the OU process, whose covariance kernel is given by

$$k(x_{\tau_1}, x_{\tau_2}) = \sigma^2 e^{-|\tau_1 - \tau_2|}, \quad \text{(Equation 7)}$$

is computationally feasible, even for long time series. While, in general, sampling from Gaussian processes scales cubically with the length of the time series, for the OU process, it is possible in linear time. This is because the OU process can be described equivalently by the stochastic differential equation

$$dx_\tau = -\rho x_\tau d\tau + \sigma dW_\tau, \quad \text{(Equation 8)}$$

where W_τ is the Wiener process. Furthermore, since the OU process is a stationary Gauss-Markov process, its precision matrix Σ^{-1} is banded, and the modified DDPM training objective can also be computed in linear time. For both sampling and computing the training objective, we apply the OU process independently for each time series channel with a fixed lengthscale ρ. However, we find that this modification does not improve sample quality in all datasets. See experiments on the effect of using the OU process versus white noise in the supplemental information.

Model training and inference

Our convolutional denoising networks are trained with stochastic gradient descent on the objective given in Equation 6. We train all our models using the AdamW optimizer[69] and choose hyperparameters by inspecting the quality of the generated data with respect to the training set. We check for overfitting by computing pairwise distances between real and generated data in the time and frequency domains. See the supplemental information for further details and network and training hyperparameters (Table S2). To sample and impute using the trained denoising network, a sample from the Gaussian distribution (white noise or OU) is gradually denoised following the standard DDPM denoising procedure.[21]

DDPMs naturally lend themselves to imputation by guiding the diffusion process based on observed data points.[21] For image-based diffusion models, this process is usually known as inpainting.[49] It works by setting the observed part of the image—or time series—to the analytical latent state of the forward diffusion process (as given by Equation 1) after each denoising step. This way, the reverse diffusion process is "guided" along the observed part of the time series. However, this approach does not provide the correct conditional distributions of the form $p(\mathbf{x}_{\text{observed}}|\mathbf{x}_{\text{missing}})$, which can hurt imputation performance.[31]

To alleviate this issue, it is possible to train conditional diffusion models: here, the observed data are given as input to the denoising network; that is, the diffusion model is trained to model the desired conditional distribution directly. In order to obtain a conditional diffusion model that can deal with arbitrary patterns of missingness at imputation time, many different possible such

patterns need to be sampled during training. In this work, we use the random sampling strategy[31] and randomly drop out channels for a given time series following a two-step process: first, we uniformly sample a number k between one and the total number of channels C and then randomly drop out k channels. The denoising network is then conditioned on the remaining $C - k$ channels, and the training objective in Equation 6 is computed. We use this conditional training strategy only for the AJILE12 dataset. For the other two datasets, the models are trained without randomly dropping out channels. All models are implemented and trained with the PyTorch library.[70]

Datasets

We perform our three experiments on three different datasets of publicly available neurophysiological recordings.

The dataset for our first experiment (Figure 2) is a three-electrode LFP recording from rats recorded at 600 Hz at the mPFC, hippocampal CA1 region, and thalamic nucleus RE.[44] It consists of a total of 3.5 h of recordings from three different rats. During the recording, rats were freely behaving with the possibility to rest and underwent cycles of sleep and wakefulness. For our experiment, we used the LFP data from the first and third recordings (52 and 93 min).

The AJILE12 dataset used for our second experiment (Figure 3) consists of human ECoG data recorded during naturalistic behavior from 12 different participants undergoing epilepsy monitoring.[45] It was recorded at 500 Hz over several days and is a total of 1,280 h. The length of recording per participant ranges from 70 to 120 h. The number of electrodes per participant ranges from 64 to 126. In addition to the neurophysiological recordings, a variety of behavioral- and movement-event-related metadata are provided.

The dataset for our third experiment (Figure 4) consists of ECoG recordings from a nonhuman primate (macaque monkey) recorded at 1,000 Hz via 128 surface electrodes placed inside the cranium, directly on top of the cortex.[52] During the recording, the macaque (Chibi) was injected with an anesthetic (propofol), and different neurophysiological states were annotated. We use the data from the anesthetized and awake-eyes-closed condition (73 min of recording).

For our initial example (Figure 1E–1H), we use data from participant P02 of the BCI Challenge @ NER 2015.[43] The data were recorded at 200 Hz with 56 passive EEG sensors placed with the extended 10–20 system. The participants were tested under the "P-300 Speller" paradigm, where words are spelled out letter by letter by flashing screen items and measuring the evoked response.[71]

Data preprocessing

For our datasets of rat LFP and macaque ECoG recordings, we perform channel-by-channel normalization by standardizing the recording over its entire duration. After normalization, we create the set of time windows that are used to train and evaluate the model. For the rat LFP recordings, we took 2-s time windows consisting of 1,200 time points, resulting in a total of 4,350 time series. Similarly, for the macaque ECoG dataset, we took 1-s time windows consisting of 1,000 time points, resulting in a total of 1,088 time series. After creating the time series, we then randomly divide them into a training and an evaluation set (80% training, 20% evaluation).

The data of each participant in the AJILE12 dataset consist of 7 days of recording. We follow the setup and preprocessing described in Peterson et al.[51]: the ECoG traces are downsampled before extracting time windows corresponding to either movement or rest. For each participant, the final dataset consists of 4-s time windows with a sampling frequency of 250 Hz, with an equal amount of time series recorded under rest and movement (i.e., class balanced). Again, we normalize the time series channel by channel by aggregating over all time series in the training set. The normalization of the training set, which consists of the first 6 days of recording, is applied to the evaluation set, that is, the seventh day of recording. Furthermore, extreme outlier channels, where the difference between the minimum and maximum values is five or more times the interquartile range, are detected and masked during training and evaluation. The number of recorded channels varies widely between participants, ranging from 64 to 126. Similarly, the number of training and evaluation time series varies considerably between participants. However, the network hyperparameters are shared between participants, and no attempt was made to fine-tune the imputation for an individual participant.

Unlike the DDPM, the random forest neural decoder is trained and evaluated on the unnormalized AJILE12 data (see Peterson et al.[51] for details). Imputations are obtained by imputing normalized time series and then reversing the normalization for both the observed and imputed channels using the normalization constants from the training data.

For our EEG dataset, we extract 1.3 s of the recording immediately following a presented stimulus. Since we only use data from participant P02, this results in 340 time series (i.e., trials) consisting of 260 time points each. After extraction, we band-pass filter the time series with a frequency window between 1 and 40 Hz using a fifth-order Butterworth filter. We then again perform channel-by-channel normalization by aggregating over the time series.

Analyses and metrics

PSDs

Throughout the experiments, we compare the distributions of (cross-)PSDs between channels of real and generated data. PSDs are computed with a Fourier transform over the whole the time series. For a given one-dimensional time series x and frequency f, the PSD is denoted by $S_{xx}(f)$. Since we are always interested in capturing the full distribution of PSDs over a set of time series, we compute the point-wise median and percentiles: here, point-wise means that, for a set of real or generated time series $\{\mathbf{x}_i\}_{i=1}^{n}$, the median $\text{median}[S_{xx}(f)]$ over all PSDs is computed at each frequency f, and similarly for the 10% and 90% percentiles.

SWR detection and cross-channel couplings

For our experiments with the rat LFP data, we detect ripples in the CA1 channel. Following the heuristic approach of Varela and Wilson,[44] we perform this detection by band-pass filtering both real and generated CA1 traces within the ripple frequency band from 100 to 275 Hz using a finite impulse response (FIR) filter. Ripples are detected when the power of the time series is more than three standard deviations greater than the mean. The mean and standard deviation of the filtered power are calculated over the entire set of real time series. Crucially, to ensure a fair comparison between a set of real and a set of generated CA1 traces, the mean and standard deviation from the real data are used to detect SWRs in the generated data. SWRs with less than 10 time points (17 ms) are fused. After fusing, an SWR must have a minimum length of 20 time points (33 ms). Detected ripples below this threshold are discarded.

In our SWR prediction experiment, we impute the CA1 channel given the other two channels and detect SWRs in the real and imputed channel for all time series in the evaluation set. If there are no SWRs or at least one SWR in both the real and generated channels, then the prediction is counted as correct. We then compute the F1 score based on the number of correct and incorrect predictions. To test if the prediction performance is significantly better than random, we shuffle the set of generated time series and evaluate the F1 score. This procedure is repeated 1,000 times to compute the permutation test statistic.[50]

We also use detected SWRs to compute a "phase-occurrence coupling" against the phase of delta (1–4 Hz), theta (4–9 Hz), and spindle (6–14 Hz) frequencies in the mPFC. The mPFC phase is calculated using the Hilbert transform after band-pass filtering (FIR, Hamming) with the corresponding frequency band. Given the mPFC phase, the time point with maximal power within a detected SWR is used to extract the phase of the SWR occurrence, and the counts are binned into 21 equally spaced phase bins (from -180° to 180°) to create the histograms (Figure 2D).

Finally, we compute the PAC between the CA1 amplitude in the ripple band (100–275 Hz) and the mPFC phase of the previously mentioned driver frequencies. Phase and amplitude in the respective frequency bands are calculated using the Hilbert transform, and the average amplitudes occurring at 31 equally spaced phase bins (from -180° to 180°) are reported (Figure 2D).

Details on the imputation experiment and neural decoder

For our imputation experiment, we follow the setup described in Talukder et al.[30]: we impute artificially dropped channels and investigate the effect of this imputation on the performance of a neural decoder. As a baseline imputation method, we use mean imputation. The neural decoding task is to discriminate between time series recorded at rest and time series recorded during movement. The neural decoder used is a random forest, which works by "flattening" a time series into a large feature vector. We train three decoders over three training folds as in Peterson et al.[51] using the same hyperparameters (maximum tree depth and number of estimators). No attempt was made to

further optimize the random forest models. Finally, the average test performance over all training folds is calculated for all settings, that is, fully observed, mean imputation, deterministic-network-based imputation, and DDPM-based imputation.

To evaluate the performance of DDPM-based imputation, we randomly select 50%, 70%, or 90% of the channels from a given participant. For each time series in the evaluation set, this set of channels is dropped and then imputed with the mean from the training set or imputed using the trained DDPM. We then apply the previously trained decoder to both imputed evaluation sets and compute the evaluation accuracy. This process is repeated 5 times for each missingness level to evaluate the imputation performance over different sets of dropped channels. We report the mean and standard deviation over the 5 repetitions (Figure 3).

Likelihood computation and brain-state classification

Likelihood computation in DDPMs can be realized by solving the probability flow equation.[72] In our class-conditional model for the macaque ECoG data, we compute log likelihoods for both the awake and anesthetized states. That is, given a time series $\mathbf{x}$, we get $\log p_\theta(\mathbf{x}|c = \text{awake})$ and $\log p_\theta(\mathbf{x}|c = \text{anesthetized})$. These values can be used to perform classification by selecting the class with the higher log likelihood for a given sample. We perform this classification for all time series in our class-balanced evaluation set and report the overall classification accuracy. For the time-resolved brain state classification (Figure 4E), we simply compute this log likelihood difference over 1-s sliding windows.

For outlier detection, we are interested in the total log likelihood $\log p_\theta(\mathbf{x})$ of sample $\mathbf{x}$. We compute this total log likelihood for each evaluation time series and correspondingly generated outliers (white noise or time-wise flipped) according to the law of total probability. Then, the AUC of the time series against both types of outliers is computed.

For the brain-state classification with synthetic data, we use the convolutional neural network classifier described in Wang et al.[53] The classifier is trained twice, once on real data and once on an equal amount of DDPM-generated data, and then evaluated on a held-out evaluation set of real data. Both the training and evaluation datasets consist of an equal number of awake and anesthetized time windows, and we compute the evaluation accuracy to compare the performance between the real and synthetic settings. The classifier is trained using the AdamW optimizer[69] and binary cross-entropy loss with identical network and training hyperparameters in both settings.

SUPPLEMENTAL INFORMATION

Supplemental information can be found online at https://doi.org/10.1016/j.patter.2024.101047.

ACKNOWLEDGMENTS

R.G. was supported by the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement no. 101030918 (AutoMIND). This work was supported by the German Research Foundation (DFG) through Germany's Excellence Strategy, EXC number 2064/1–390727645 and SFB1233 (PN 276693517); the German Federal Ministry of Education and Research (BMBF): Tübingen AI Center, FKZ: 01IS18039A; and the European Union (ERC, DeepCoMechTome, 101089288). J.V. is a member of the International Max Planck Research School for Intelligent Systems (IMPRS-IS) and the AI4Med-BW graduate program. We would like to thank Auguste Schulz and Matthijs Pals for feedback on the manuscript and Mackelab members for discussions.

AUTHOR CONTRIBUTIONS

Conceptualization, J.V., J.H.M., and R.G.; formal analysis, J.V.; methodology, J.V.; visualization, J.V.; software, J.V.; supervision, J.H.M. and R.G.; writing – original draft, J.V.; writing – review & editing, J.V., J.H.M., and R.G.; funding acquisition, J.H.M.

DECLARATION OF INTERESTS

The authors declare no competing interests.

Received: April 6, 2024
Revised: July 1, 2024
Accepted: July 31, 2024
Published: August 29, 2024

REFERENCES

1. Krumin, M., and Shoham, S. (2009). Generation of spike trains with controlled auto-and cross-correlation functions. Neural Comput. *21*, 1642–1664.
2. Macke, J.H., Berens, P., Ecker, A.S., Tolias, A.S., and Bethge, M. (2009). Generating spike trains with specified correlation coefficients. Neural Comput. *21*, 397–423.
3. Gutnisky, D.A., and Josić, K. (2010). Generation of spatiotemporally correlated spike trains and local field potentials using a multivariate autoregressive process. J. Neurophysiol. *103*, 2912–2930.
4. Schreiber, T., and Schmitz, A. (2000). Surrogate time series. Phys. Nonlinear Phenom. *142*, 346–382.
5. Venema, V., Ament, F., and Simmer, C. (2006). A stochastic iterative amplitude adjusted Fourier transform algorithm with improved accuracy. Nonlinear Process Geophys. *13*, 321–328.
6. Gifford, A.T., Dwivedi, K., Roig, G., and Cichy, R.M. (2022). A large and rich EEG dataset for modeling human visual object recognition. Neuroimage *264*, 119754.
7. Kong, Z., Ping, W., Huang, J., Zhao, K., and Catanzaro, B. (2021). Diffwave: A versatile diffusion model for audio synthesis. International Conference on Learning Representations *abs/2009.09761*. https://doi.org/10.48550/arXiv.2009.09761.
8. Dhariwal, P., and Nichol, A. (2021). Diffusion models beat GANs on image synthesis. Adv. Neural Inf. Process. Syst. *34*, 8780–8794.
9. Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. (2022). High-resolution image synthesis with latent diffusion models. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 10684–10695.
10. Fortuin, V., Baranchuk, D., Rätsch, G., and Mandt, S. (2020). GP-VAE: Deep probabilistic time series imputation. International Conference on Artificial Intelligence and Statistics, 1651–1661. https://doi.org/10.48550/arXiv.1907.04155.
11. Sanchez-Lengeling, B., and Aspuru-Guzik, A. (2018). Inverse molecular design using machine learning: Generative models for matter engineering. Science *361*, 360–365.
12. Sun, H., and Bouman, K.L. (2021). Deep probabilistic imaging: Uncertainty quantification and multi-modal solution characterization for computational imaging. Proc. AAAI Conf. Artif. Intell. *35*, 2628–2637.
13. Pandarinath, C., O'Shea, D.J., Collins, J., Jozefowicz, R., Stavisky, S.D., Kao, J.C., Trautmann, E.M., Kaufman, M.T., Ryu, S.I., Hochberg, L.R., et al. (2018). Inferring single-trial neural population dynamics using sequential auto-encoders. Nat. Methods *15*, 805–815.
14. Molano-Mazon, M., Onken, A., Piasini, E., and Panzeri, S. (2018). Synthesizing realistic neural population activity patterns using generative adversarial networks. International Conference on Learning Representations. https://doi.org/10.48550/arXiv.1803.00338.
15. Ramesh, P., Atayi, M., and Macke, J.H. (2019). Adversarial training of neural encoding models on population spike trains. Real Neurons & Hidden Units: Future directions at the intersection of neuroscience and artificial intelligence @ NeurIPS 2019.
16. Aznan, N.K.N., Atapour-Abarghouei, A., Bonner, S., Connolly, J.D., Al Moubayed, N., and Breckon, T.P. (2019). Simulating brain signals: Creating synthetic eeg data via neural-based generative models for improved ssvep classification. In 2019 International joint conference on neural networks (IJCNN), pp. 1–8.
17. Hartmann, K.G., Schirrmeister, R.T., and Ball, T. (2018). Eeg-gan: Generative adversarial networks for electroencephalograhic (eeg) brain signals. Preprint at arXiv. https://doi.org/10.48550/arXiv.1806.01875.

18. Luo, Y., Zhu, L.-Z., Wan, Z.-Y., and Lu, B.-L. (2020). Data augmentation for enhancing eeg-based emotion recognition with deep generative models. J. Neural. Eng. *17*, 056021.

19. Ponce, C.R., Xiao, W., Schade, P.F., Hartmann, T.S., Kreiman, G., and Livingstone, M.S. (2019). Evolving images for visual neurons using a deep generative network reveals coding principles and neuronal preferences. Cell *177*, 999–1009.e10.

20. Lin, S., Sprague, T., and Singh, A.K. (2022). Mind reader: Reconstructing complex images from brain activities. Adv. Neural Inf. Process. Syst. *35*, 29624–29636.

21. Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models. Adv. Neural Inf. Process. Syst. *33*, 6840–6851.

22. Yang, L., Zhang, Z., Song, Y., Hong, S., Xu, R., Zhao, Y., Zhang, W., Cui, B., and Yang, M.-H. (2023). Diffusion models: A comprehensive survey of methods and applications. ACM Comput. Surv. *56*, 1–39.

23. Lin, L., Li, Z., Li, R., Li, X., and Gao, J. (2024). Diffusion models for time series applications: A survey. Front. Inform. Technol. Electron. Eng. *25*, 19–41.

24. Takagi, Y., and Nishimoto, S. (2023). High-resolution image reconstruction with latent diffusion models from human brain activity. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 14453–14463. https://doi.org/10.1101/2022.11.18.517004.

25. Chen, Z., Qing, J., Xiang, T., Yue, W.L., and Zhou, J.H. (2023). Seeing beyond the brain: Conditional diffusion model with sparse masked modeling for vision decoding. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 22710–22720. https://doi.org/10.48550/arXiv.2211.06956.

26. Silva, I., Moody, G., Scott, D.J., Celi, L.A., and Mark, R.G. (2012). Predicting in-hospital mortality of icu patients: The physionet/computing in cardiology challenge 2012. Comput. Cardiol. *39*, 245–248.

27. Walonoski, J., Kramer, M., Nichols, J., Quina, A., Moesel, C., Hall, D., Duffett, C., Dube, K., Gallagher, T., and McLachlan, S. (2018). Synthea: An approach, method, and software mechanism for generating synthetic patients and the synthetic electronic health care record. J. Am. Med. Inform. Assoc. *25*, 230–238.

28. Yoon, J., Jarrett, D., and Van der Schaar, M. (2019). Time-series generative adversarial networks. Adv. Neural Inf. Process. Syst. *32*, 5509–5519.

29. Abbasi, S., Maran, S., and Jaeger, D. (2020). A general method to generate artificial spike train populations matching recorded neurons. J. Comput. Neurosci. *48*, 47–63.

30. Talukder, S., Sun, J.J., Leonard, M., Brunton, B.W., and Yue, Y. (2022). Deep neural imputation: A framework for recovering incomplete brain recordings. NeurIPS 2022 Workshop on Learning from Time Series for Health. https://doi.org/10.48550/arXiv.2206.08094.

31. Tashiro, Y., Song, J., Song, Y., and Ermon, S. (2021). CSDI: Conditional score-based diffusion models for probabilistic time series imputation. Adv. Neural Inf. Process. Syst. *34*, 24804–24816.

32. Alcaraz, J.L., and Strodthoff, N. (2022). Diffusion-based time series imputation and forecasting with structured state space models. Transactions on Machine Learning Research. https://doi.org/10.48550/arXiv.2208.09399.

33. Rasul, K., Seward, C., Schuster, I., and Vollgraf, R. (2021). Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting. International Conference on Machine Learning, 8857–8868. https://doi.org/10.48550/arXiv.2101.12072.

34. Biloš, M., Rasul, K., Schneider, A., Nevmyvaka, Y., and Günnemann, S. (2023). Modeling temporal data as continuous functions with process diffusion. International Conference on Machine Learning *202*, 2452–2470.

35. Shu, K., Zhao, Y., Wu, L., Liu, A., Qian, R., and Chen, X. (2023). Data augmentation for seizure prediction with generative diffusion model. Preprint at arXiv. https://doi.org/10.48550/arXiv.2306.08256.

36. Li, Y., Cai, T., Zhang, Y., Chen, D., and Dey, D. (2023). What makes convolutional models great on long sequence modeling? International Conference on Learning Representations *abs/2210.09298*. https://doi.org/10.48550/arXiv.2210.09298.

37. Gu, A., Goel, K., and Ré, C. (2022). Efficiently modeling long sequences with structured state spaces. International Conference on Learning Representations *abs/2111.00396*. https://doi.org/10.48550/arXiv.2111.00396.

38. Smith, J.T., Warrington, A., and Linderman, S.W. (2023). Simplified state space layers for sequence modeling. International Conference on Learning Representations *abs/2208.04933*. https://doi.org/10.48550/arXiv.2208.04933.

39. Pritchard, W.S. (1992). The brain in fractal time: 1/f-like power spectrum scaling of the human electroencephalogram. Int. J. Neurosci. *66*, 119–129.

40. He, B.J., Zempel, J.M., Snyder, A.Z., and Raichle, M.E. (2010). The temporal structures and functional significance of scale-free brain activity. Neuron *66*, 353–369.

41. Gao, R., Peterson, E.J., and Voytek, B. (2017). Inferring synaptic excitation/inhibition balance from field potentials. Neuroimage *158*, 70–78.

42. Donoghue, T., Haller, M., Peterson, E.J., Varma, P., Sebastian, P., Gao, R., Noto, T., Lara, A.H., Wallis, J.D., Knight, R.T., et al. (2020). Parameterizing neural power spectra into periodic and aperiodic components. Nat. Neurosci. *23*, 1655–1665.

43. Margaux, P., Emmanuel, M., Sébastien, D., Olivier, B., and Jérémie, M. (2012). Objective and subjective evaluation of online error correction during P300-based spelling. Advances in Human-Computer Interaction *2012*, 578295. https://doi.org/10.1155/2012/578295.

44. Varela, C., and Wilson, M.A. (2020). mPFC spindle cycles organize sparse thalamic activation and recently active CA1 cells during non-REM sleep. Elife *9*, e48881.

45. Peterson, S.M., Singh, S.H., Dichter, B., Scheid, M., Rao, R.P.N., and Brunton, B.W. (2022). AJILE12: Long-term naturalistic human intracranial neural recordings and pose. Sci. Data *9*, 184.

46. Varela, C., and Wilson, M.A. (2019). Simultaneous extracellular recordings from midline thalamic nuclei, medial prefrontal cortex and CA1 from rats cycling through bouts of sleep and wakefulness. CRCNS.org. https://doi.org/10.6080/K0K35RVG.

47. Kingma, D.P., and Welling, M. (2013). Auto-encoding variational bayes. International Conference on Learning Representations. https://doi.org/10.48550/arXiv.1312.6114.

48. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2020). Generative adversarial networks. Commun. ACM *63*, 139–144.

49. Lugmayr, A., Danelljan, M., Romero, A., Yu, F., Timofte, R., and Van Gool, L. (2022). Repaint: Inpainting using denoising diffusion probabilistic models. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 11461–11471.

50. Ojala, M., and Garriga, G.C. (2010). Permutation tests for studying classifier performance. J. Mach. Learn. Res. *11*.

51. Peterson, S.M., Steine-Hanson, Z., Davis, N., Rao, R.P.N., and Brunton, B.W. (2021). Generalized neural decoders for transfer learning across participants and recording modalities. J. Neural. Eng. *18*, 026014.

52. Yanagawa, T., Chao, Z.C., Hasegawa, N., and Fujii, N. (2013). Large-scale information flow in conscious and unconscious states: An ECoG study in monkeys. PLoS One *8*, e80845.

53. Wang, Z., Yan, W., and Oates, T. (2017). Time series classification from scratch with deep neural networks: A strong baseline. International joint conference on neural networks (IJCNN), 1578–1585. https://doi.org/10.48550/arXiv.1611.06455.

54. Xu, X., Kong, Q., Zhang, D., and Zhang, Y. (2024). An evaluation of inter-brain eeg coupling methods in hyperscanning studies. Cogn. Neurodyn. *18*, 67–83.

55. Burgess, A.P. (2013). On the interpretation of synchronization in eeg hyperscanning studies: a cautionary note. Front. Hum. Neurosci. *7*, 881.

56. Song, J., Meng, C., and Ermon, S. (2021a). Denoising diffusion implicit models. International Conference on Learning Representations *abs/2010.02502*. https://doi.org/10.48550/arXiv.2010.02502.

57. Karras, T., Aittala, M., Aila, T., and Laine, S. (2022). Elucidating the design space of diffusion-based generative models. Adv. Neural Inf. Process. Syst. *35*, 26565–26577.

58. Lu, C., Zhou, Y., Bao, F., Chen, J., Li, C., and Zhu, J. (2022). DPM-Solver: A fast ODE solver for diffusion probabilistic model sampling in around 10 steps. Adv. Neural Inf. Process. Syst. *35*, 5775–5787.

59. Vyas, S., Golub, M.D., Sussillo, D., and Shenoy, K.V. (2020). Computation through neural population dynamics. Annu. Rev. Neurosci. *43*, 249–275.

60. Gallego-Carracedo, C., Perich, M.G., Chowdhury, R.H., Miller, L.E., and Gallego, J.Á. (2022). Local field potentials reflect cortical population dynamics in a region-specific and frequency-dependent manner. Elife *11*, e73155.

61. Chung, H., Kim, J., Mccann, M.T., Klasky, M.L., and Ye, J.C. (2023). Diffusion posterior sampling for general noisy inverse problems. International Conference on Learning Representations. https://doi.org/10.48550/arXiv.2209.14687.

62. van den Burg, G., and Williams, C. (2021). On memorization in probabilistic deep generative models. Adv. Neural Inf. Process. Syst. *34*, 27916–27928.

63. Vetter, J., Macke, J.H., and Gao, R. (2024). Code for the Article "Generating Realistic Neurophysiological Time Series with Denoising Diffusion Probabilistic Models (Zenodo). https://doi.org/10.5281/zenodo.12759366.

64. Peebles, W., and Xie, S. (2023). Scalable diffusion models with transformers. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 4195–4205. https://doi.org/10.48550/arXiv.2212.09748.

65. LeCun, Y., Bengio, Y., et al. (1995). Convolutional networks for images, speech, and time series. The handbook of brain theory and neural networks *3361*, 255–258.

66. Hendrycks, D., and Gimpel, K. (2016). Gaussian error linear units. Preprint at arXiv. https://doi.org/10.48550/arXiv.1606.08415.

67. Ioffe, S., and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. International Conference on Machine Learning, 448–456. https://doi.org/10.48550/arXiv.1502.03167.

68. Ba, J.L., Kiros, J.R., and Hinton, G.E. (2016). Layer normalization. Preprint at arXiv:1607.06450. https://doi.org/10.48550/arXiv.1607.06450.

69. Loshchilov, I., and Hutter, F. (2019). Decoupled weight decay regularization. International Conference on Learning Representations. https://doi.org/10.48550/arXiv.1711.05101.

70. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. Adv. Neural Inf. Process. Syst. *32*.

71. Farwell, L.A., and Donchin, E. (1988). Talking off the top of your head: Toward a mental prosthesis utilizing event-related brain potentials. Electroencephalogr. Clin. Neurophysiol. *70*, 510–523.

72. Song, Y., Sohl-Dickstein, J., Kingma, D.P., Kumar, A., Ermon, S., and Poole, B. (2021b). Score-based generative modeling through stochastic differential equations. International Conference on Learning Representations. https://doi.org/10.48550/arXiv.2011.13456.

Patterns, Volume *5*

Supplemental information

Generating realistic neurophysiological time series with denoising diffusion probabilistic models

Julius Vetter, Jakob H. Macke, and Richard Gao

Supplemental Information

Supplemental Experimental Procedures

Ornstein-Uhlenbeck process as diffusion process

In our experiments, we sometimes use the OU process instead of the classically used white noise as our diffusion process. This substitution is possible because the OU process belongs to the family of Gaussian processes. See Biloš et al.[1] for more details.

The training and sampling schemes are shown in Algorithm 1 and Algorithm 2 and differ slightly from those given in Biloš et al.[1]. Note that the extension to general Gaussian processes has a computational overhead compared to the original training objective. During training, the computation of the Mahalanobis distance scales quadratically with the length of the input. Sampling from a Gaussian process also requires computing a Cholesky decomposition. However, this decomposition only needs to be computed once, and can also be used to compute the Mahalanobis efficiently by solving a lower triangular linear system based on the Cholesky decomposition.

Algorithm 1 Training

$L = \mathrm{Cholesky}(\Sigma)$
repeat
$\mathbf{x}_0 \sim q(\mathbf{x}_0)$
$t \sim \mathrm{Uniform}(\{1, \dots, T\})$
$\boldsymbol{\epsilon} \sim \mathcal{N}(0, \Sigma)$
Take gradient descent step on
$\quad \nabla_\theta(\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t))^T \Sigma^{-1}(\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t))$
until converged

Algorithm 2 Sampling

$\mathbf{x}_T \sim \mathcal{N}(0, \Sigma)$
for $t = T, \dots, 1$ **do**
$\quad z \sim \mathcal{N}(0, \Sigma)$ if $t > 1$, else $z = 0$
$\quad \mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)) + \sigma_t z$
end for
return $\mathbf{x}_0$

As described in the main text, the computational requirements scale more favorably for the special case of the OU process. Its formulation as a stochastic differential equation allows sampling in linear time. Furthermore, the OU process has a tridiagonal precision matrix, which allows computing the training objective in linear time as well.

Additional results and details

Here we provide additional results and details. The hyperparameters used for our experiments are given in Table S2.

For completeness, all distributions of PSDs of AJILE12 participant P07 in the main text are given (Fig. S7). Additionally, all distributions of PSDs for two other participants, participant P01 with 94 channels (Fig. S8) and participant P12 with 126 channels (Fig. S9) are provided. In both cases, the median spectra as well as 10 % and 90 % percentiles match well. Recall that the neural decoding performance on data imputed by our model did not improve over the mean imputation baseline for participant P01. Nevertheless, the overall spectra of the real and imputed data match. This means that while the model was able to capture the overall distribution of time series, it likely failed to learn the class-conditional mapping between observed and missing channels. The average correlations between ground truth and imputations are positive for all 12 participants. With increasing levels of dropout, they drop towards zero (Table S1). The relative improvements of the DDPM-based imputations over the mean imputation baseline are on average better than the ones obtained by the deterministic CNNAE model (Talukder et al.[2], Table S1). We also provide an example of a deterministic, neural network-based imputation (Fig. S5).

As an additional experiment with high channel count, we generated time series from the awake condition of the macaque ECoG data on all 128 recorded channels. The distribution of real and generated PSDs closely match across all channels, but our model sometimes overestimates the power in the low-frequency region (Fig. S6).

Distances between real and generated data For all datasets, we generate the same number of samples as were used to train the corresponding models. We then compute all nearest-neighbor distances within the training data and from the generated samples to the training samples. Two distance measures are used, the L_1 distance in

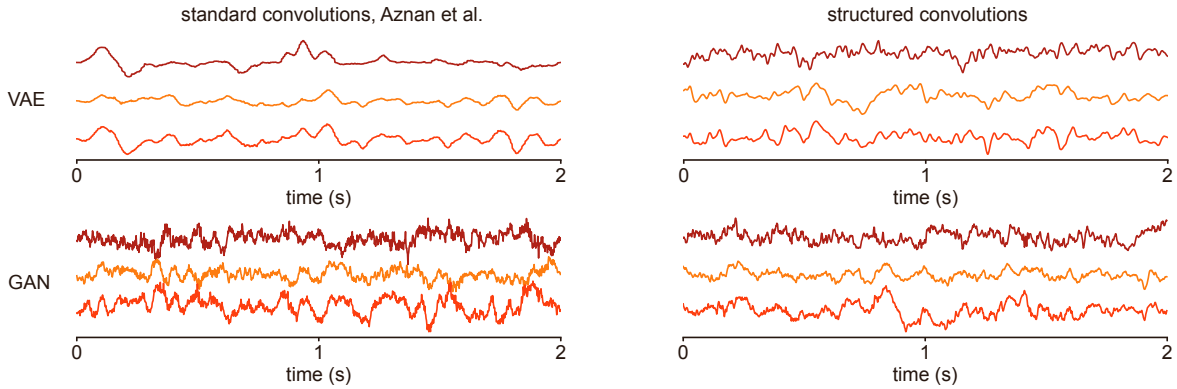


Figure S1: **Samples from GANs and VAEs with architectures based on standard and structured convolutions** The encoder/decoder and generator/discriminator architectures based on structured convolutions produce better-looking samples.

the time domain (which is small if and only if two time series are nearly exact copies of each other) and the Fourier-Wasserstein distance[3]. The Fourier-Wasserstein distance, a pseudo-metric that operates in the spectral domain, is based on normalized PSDs and will be small if, for example, two time series are scaled or translated versions of each other.

We then compare the distribution of nearest neighbor distances between the real and generated data with the distribution of distances within the training data (Fig. S4). Crucially, the minimum nearest-neighbor distance between and within the real and generated time series is away from zero for both the L_1 and the Fourier-Wasserstein distance and similar to the distribution of distances within. For a heavily overfitted DDPM, we would expect large parts of the distribution of nearest-neighbor distances between the real and generated data to be substantially smaller.

OU process versus white noise Our use of the OU process provides a modification that can further improve the quality of generated samples in some cases. Here, we provide an analysis for the AJILE12 dataset. We retrain the models using the standard independent Gaussian process (white noise). All other training and network hyperparameters are kept fixed.

For all AJILE12 participants (Fig. S10, Fig. S11, Fig. S12 for AJILE12 participants P01, P07, P12, respectively), the median power-spectra are worse using the white noise instead of the OU process, especially in the low-power, high-frequency region, which is overestimated when white noise is used.

However, for the rat LFP, the awake/anesthetized macaque ECoG data, and the BCI EEG data, there was no substantial advantage to using the OU process over white noise.

VAE and GAN baseline Here, we use the unconditional rat LFP generation experiment to establish a baseline using other types of generative models. We train VAEs and GANs using the architectures from Aznan et al.[4] as well as our own encoder/decoder and generator/discriminator architectures based on structured convolutions. The convolution hyperparameters were informed by those used in DDPM, but further tuned for the generative model at hand.

We find that the architectures used in Aznan et al.[4] are not sufficient to faithfully capture the statistics of the LFP recordings. Our architectures based on structured convolutions produce more realistic samples for both VAE and GAN (Fig. S1). However, and as we discuss below, they still fall short of the performance achieved by the DDPM:

The VAE with the encoder/decoder based on structured convolutions produces realistic but too smooth LFPs (Fig. S2A). The marginals between real and generated recordings closely match (Fig. S2B). At low frequencies, the VAE-generated samples capture the spectrum of the recordings well. At higher frequencies, however, their power drops off too quickly (Fig. S2C). As a result, the generated samples appear smooth or blurred. This is a common problem with VAEs[5,6]. Because of their smoothness, the samples do not have the correct phase-amplitude coupling (Fig. S2D).

The GAN with the generator/discriminator based on structured convolutions is able to produce realistic looking LFP recordings (Fig. S2A) whose power spectra and marginals match (Fig. S2B and C). Furthermore, we observe a non-trivial phase-amplitude coupling in the generated samples (Fig. S2D). Overall, however, the quality of the generated samples is inferior to that of the DDPM: The median spectra show strong oscillations at higher frequencies, and the couplings only approximately match those of the real data. Furthermore, we found the training of the GAN

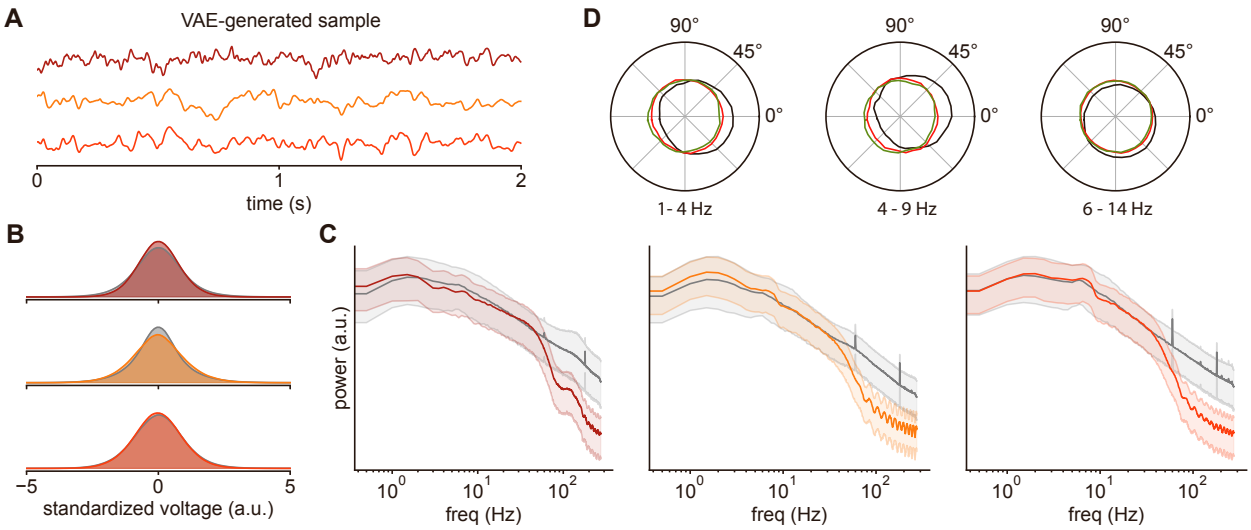


Figure S2: **VAE with structured convolutions for rat cortical-hippocampal LFP. (A)** Example of time series generated by the VAE. **(B)** Marginal distributions over the standardized voltage of each channel for both real and generated data. **(C)** Median and 10 %/90 % percentiles of real and generated power spectra for each channel. **(D)** Phase-amplitude and phase-count coupling of CA1 ripples for different driver frequencies in the mPFC for real, surrogate, and VAE-generated data.

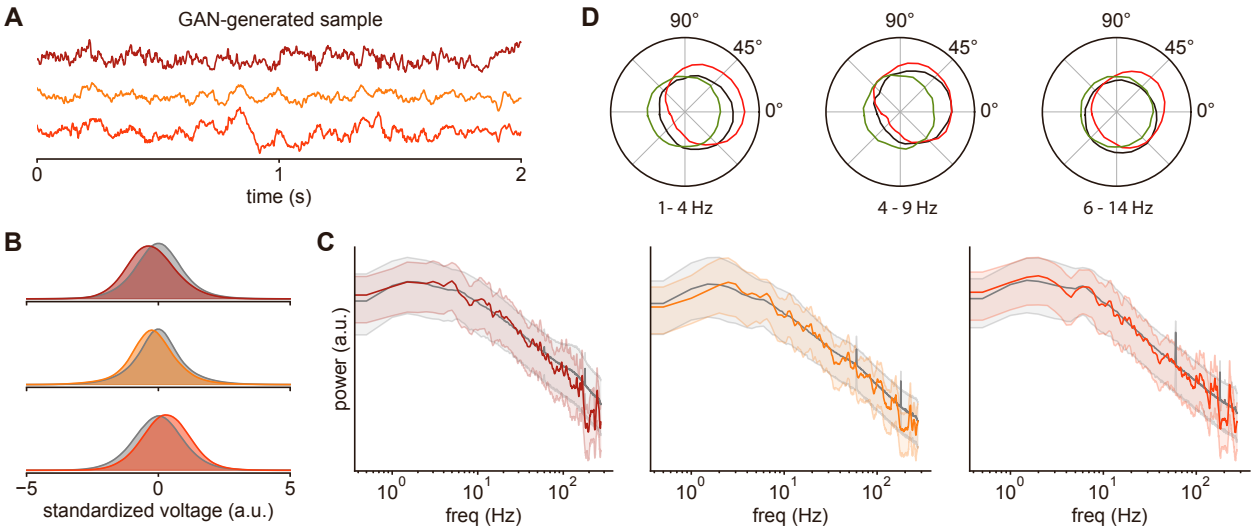


Figure S3: **GAN with structured convolutions for rat cortical-hippocampal LFP. (A)** Example of time series generated by the GAN. **(B)** Marginal distributions over the standardized voltage of each channel for both real and generated data. **(C)** Median and 10 %/90 % percentiles of real and generated power spectra for each channel. **(D)** Phase-amplitude and phase-count coupling of CA1 ripples for different driver frequencies in the mPFC for real, surrogate, and GAN-generated data.

to be relatively unstable[7,8]. Different seeds can lead to very different results, even if the chosen hyperparameters are kept fixed.

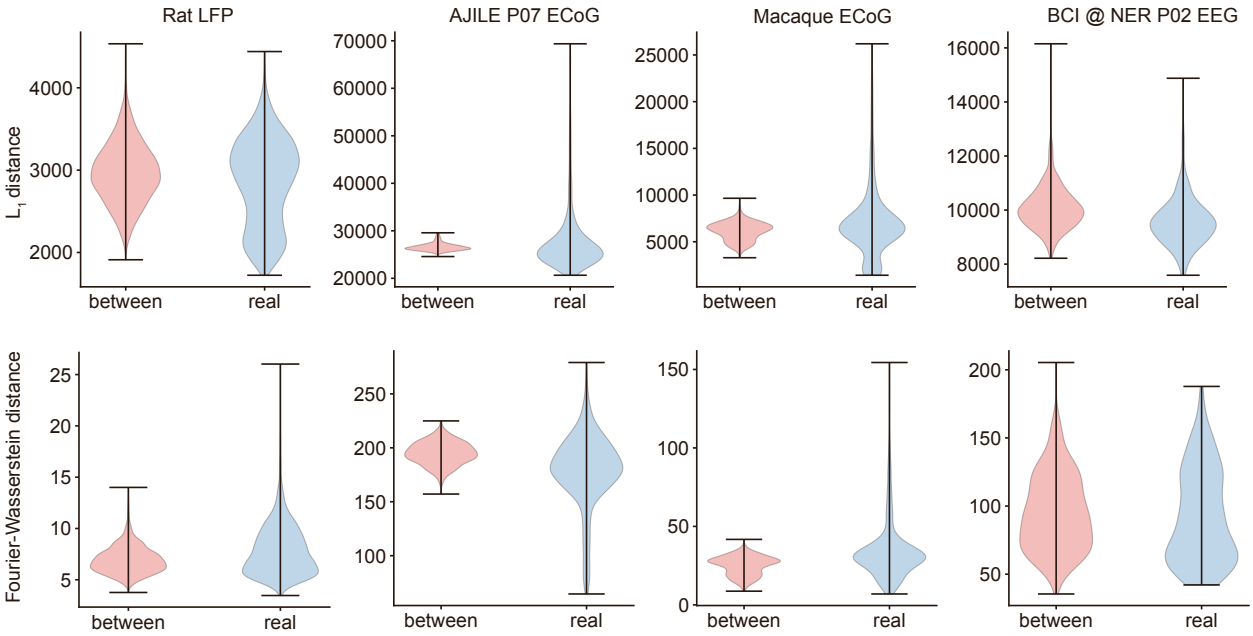


Figure S4: **Distribution of nearest-neighbor distances.** Nearest neighbors were computed from all generated to all real training time series, as well as within all training time series. The whiskers show the minimum and maximum distance (L_1 or the Fourier-Wasserstein distance). The minimum nearest-neighbor distance between real and generated time series is away from zero and similar to or larger than the minimum distance within the set of training time series.

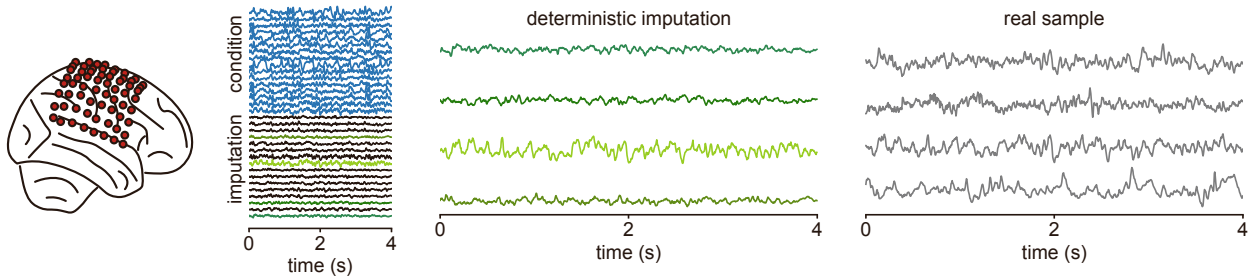


Figure S5: **Example of deterministic neural network-based imputation for the AJILE12 dataset.** The imputation shown is for participant P07. The deterministic imputation appears too flat when compared to the real sample.

Table S1: **Correlations and decoding accuracies for AJILE12 experiment.** Average correlation between imputed and ground truth channels for all test time series across 5 different dropout seeds for diffusion and neural network based imputations. Number in brackets is standard deviation over different dropout seeds. Relative improvement of decoding accuracy over mean imputation in percentage points (pp) when using either DDPM, neural-network or CNNAE imputations[2]. The average and standard deviation over 5 different dropout seeds are shown. First three rows contain the average across all participants.

Pt.	Drop.	Correlation (DDPM)	Correlation (deterministic)	Imp. in pp (DDPM)	Imp. in pp (deterministic)	Imp. in pp Talukder et al.[2]
AVG.	50 %	0.27	0.50	5.9	3.6	3.2
	70 %	0.21	0.45	7.4	6.1	4.6
	90 %	0.10	0.32	4.0	2.5	2.7
P01	50 %	0.37 (0.025)	0.54 (0.026)	−4.6 (5.69)	−10.8 (13.37)	0.2 (0.34)
	70 %	0.28 (0.011)	0.47 (0.013)	0.1 (0.07)	0.0 (0.0)	−0.2 (0.44)
	90 %	0.12 (0.021)	0.32 (0.034)	0.0 (0.04)	0.0 (0.04)	−0.2 (0.39)
P02	50 %	0.42 (0.024)	0.61 (0.020)	2.3 (2.99)	3.1 (2.86)	−1.3 (1.27)
	70 %	0.37 (0.014)	0.58 (0.009)	3.9 (4.55)	5.7 (2.82)	0.5 (2.17)
	90 %	0.18 (0.062)	0.43 (0.049)	4.1 (5.28)	3.9 (2.40)	0.3 (1.08)
P03	50 %	0.30 (0.016)	0.54 (0.015)	8.8 (4.58)	3.9 (1.43)	4.3 (2.66)
	70 %	0.24 (0.006)	0.49 (0.011)	13.7 (3.13)	6.2 (3.69)	4.8 (2.91)
	90 %	0.10 (0.027)	0.32 (0.038)	4.8 (4.07)	0.6 (0.69)	3.6 (4.26)
P04	50 %	0.19 (0.019)	0.38 (0.012)	2.8 (1.88)	6.1 (2.74)	5.0 (2.40)
	70 %	0.15 (0.012)	0.36 (0.022)	4.4 (1.92)	8.8 (1.41)	6.7 (1.81)
	90 %	0.06 (0.012)	0.23 (0.026)	1.0 (1.12)	1.9 (1.93)	1.5 (1.49)
P05	50 %	0.15 (0.007)	0.41 (0.007)	6.8 (2.04)	3.7 (1.02)	2.5 (1.30)
	70 %	0.12 (0.005)	0.37 (0.009)	7.2 (2.24)	3.6 (0.97)	3.1 (0.28)
	90 %	0.05 (0.004)	0.26 (0.011)	2.4 (1.81)	1.2 (0.56)	1.8 (0.88)
P06	50 %	0.11 (0.014)	0.37 (0.020)	0.1 (6.27)	0.6 (3.02)	2.4 (2.37)
	70 %	0.08 (0.009)	0.34 (0.005)	1.8 (3.76)	3.6 (1.22)	1.8 (1.56)
	90 %	0.03 (0.004)	0.23 (0.006)	0.2 (1.11)	−2.2 (3.01)	2.8 (1.33)
P07	50 %	0.35 (0.018)	0.59 (0.015)	14.0 (3.70)	10.6 (3.51)	9.5 (4.77)
	70 %	0.27 (0.025)	0.53 (0.016)	20.7 (1.37)	16.2 (2.68)	13.5 (3.05)
	90 %	0.13 (0.031)	0.38 (0.037)	14.5 (5.23)	7.6 (5.37)	6.7 (1.63)
P08	50 %	0.17 (0.015)	0.43 (0.018)	4.4 (4.37)	2.4 (3.51)	0.6 (2.42)
	70 %	0.13 (0.018)	0.39 (0.021)	2.5 (4.52)	5.7 (1.79)	3.0 (6.00)
	90 %	0.04 (0.007)	0.24 (0.019)	2.8 (1.38)	6.6 (1.03)	1.8 (0.47)
P09	50 %	0.27 (0.021)	0.52 (0.010)	6.5 (2.09)	3.2 (1.51)	−5.7 (1.84)
	70 %	0.21 (0.010)	0.47 (0.010)	5.9 (3.20)	3.5 (1.08)	0.8 (2.01)
	90 %	0.08 (0.012)	0.32 (0.013)	6.7 (2.39)	2.7 (1.92)	1.0 (1.17)
P10	50 %	0.30 (0.011)	0.53 (0.013)	16.1 (15.37)	8.1 (13.57)	10.2 (2.69)
	70 %	0.23 (0.018)	0.47 (0.022)	9.3 (12.89)	3.0 (9.75)	4.7 (5.57)
	90 %	0.14 (0.008)	0.36 (0.004)	1.6 (1.14)	0.5 (0.28)	0.7 (1.31)
P11	50 %	0.28 (0.014)	0.52 (0.016)	4.5 (4.97)	5.2 (5.04)	7.0 (6.97)
	70 %	0.23 (0.018)	0.47 (0.016)	9.8 (5.04)	8.3 (5.09)	11.8 (5.59)
	90 %	0.12 (0.014)	0.34 (0.018)	6.4 (5.45)	5.3 (3.63)	10.1 (7.89)
P12	50 %	0.28 (0.016)	0.55 (0.018)	8.6 (2.68)	8.0 (0.59)	3.5 (1.65)
	70 %	0.22 (0.011)	0.51 (0.011)	10.0 (2.33)	8.6 (1.57)	4.9 (0.84)
	90 %	0.11 (0.019)	0.37 (0.020)	3.6 (1.77)	1.8 (0.47)	1.7 (1.20)

Table S2: **Hyperparameters used to train models.** For the AJILE dataset, he same hyperparameters were used for all 12 participants, with the exception of the number of training epochs, to account for the large differences in training data size between participants

Hyperparameter	**LFP (rat)**	**AJILE12**	**ECoG (macaque)**	**EEG BCI**
In kernel size	32	1	32	65
Out kernel size	32	1	32	65
SC layers	3	3	3	3
SC kernel size	32	53	101	65
SC scales	5	4	1	1
Latent dim. per channel	64	16	32	16
Off-diag. size	64	4	4	8
Noise process	White noise	OU ($\rho = 10$)	White noise	White noise
Diffusion steps T	500	50	500	500
Learning rate	0.0001	0.0005	0.0004	0.0004
Weight decay	0.01	0.01	0.01	0.01
Train batch size	32	32	32	32
Epochs	500	250/500/1000	500	1000

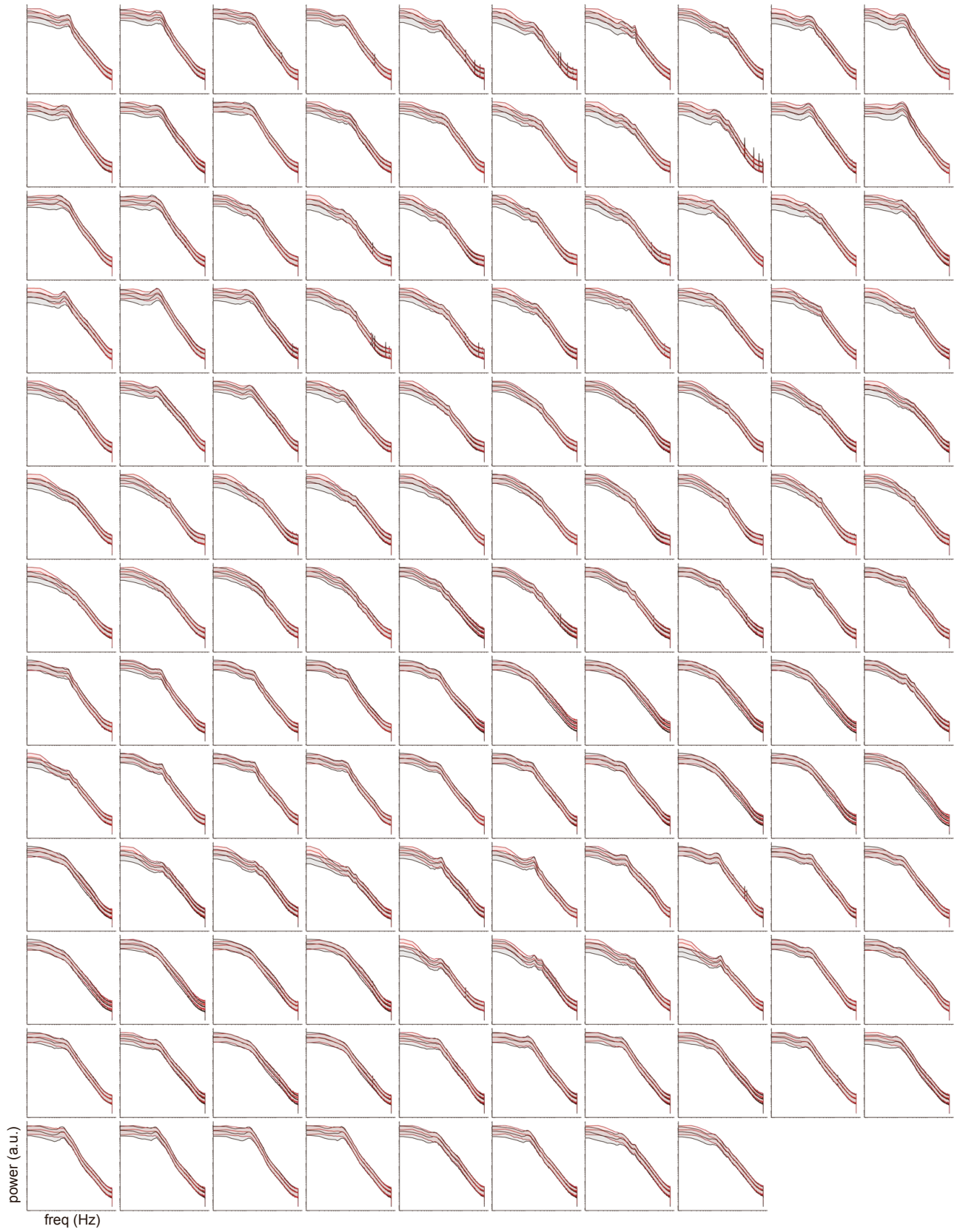


Figure S6: **Full spectra of 128 channel macaque data in awake condition.** Median power as well as 10 %/90 % percentiles are shown.

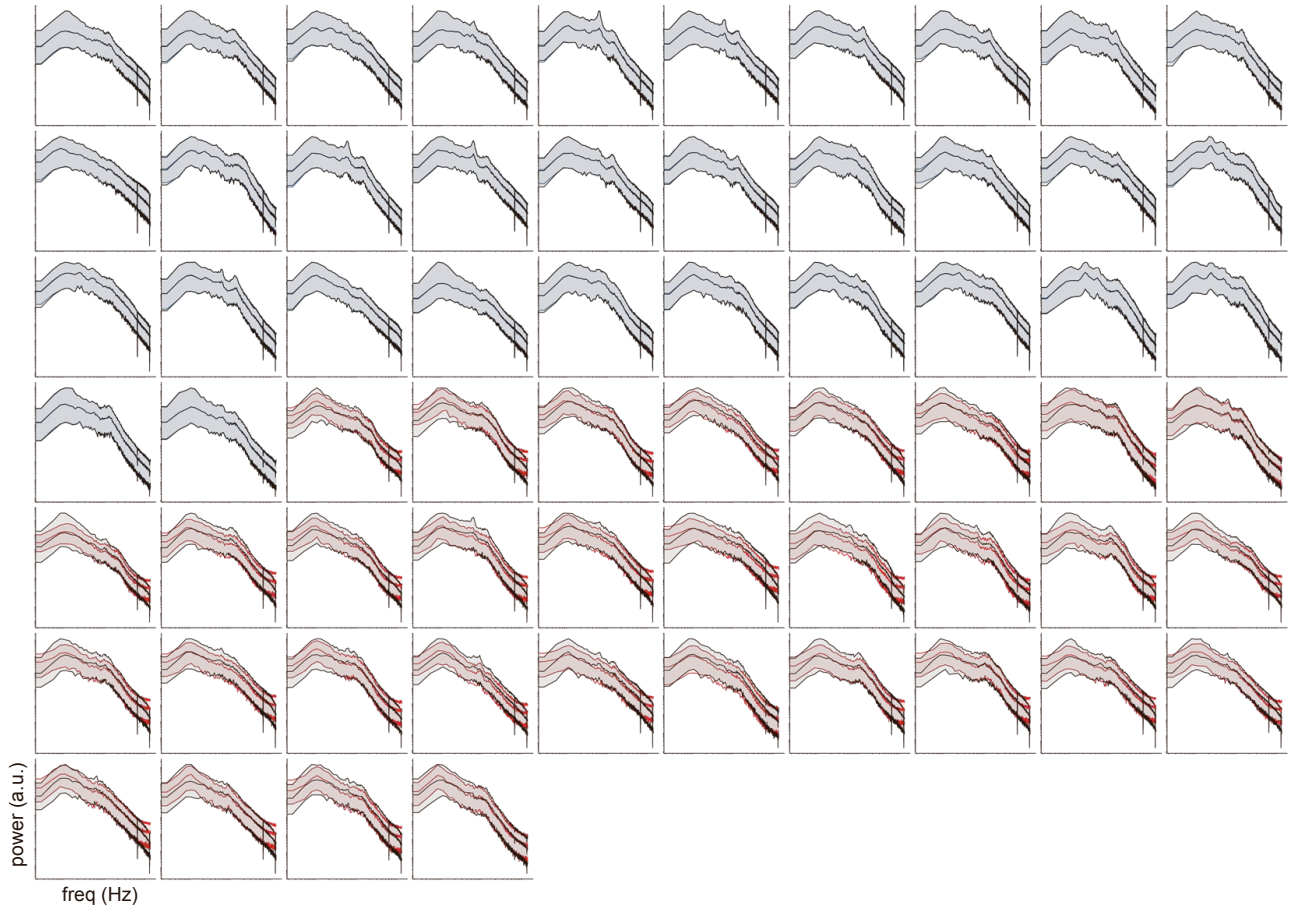


Figure S7: **Full spectra of P07 from the AJILE12 dataset.** The LFP traces consist of 64 channels. Here, we imputed the second half of the channels given the first half for all time series in the evaluation set. Median power as well as 10 %/90 % percentiles are shown.

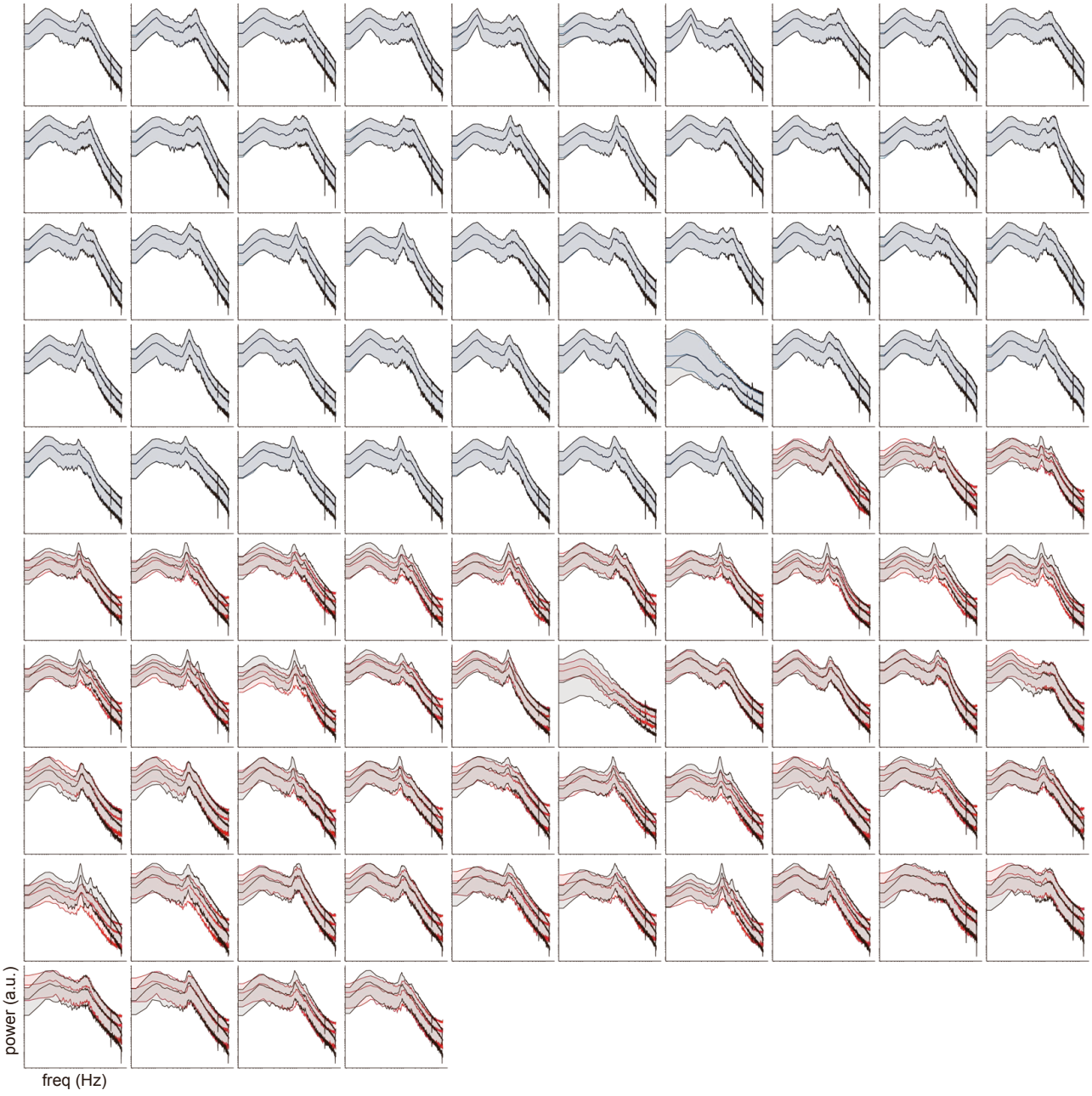


Figure S8: **Full spectra of P01 from the AJILE12 dataset.** The LFP traces consist of 94 channels. Here, we imputed the second half of the channels given the first half for all time series in the evaluation set. Median power as well as 10 %/90 % percentiles are shown.

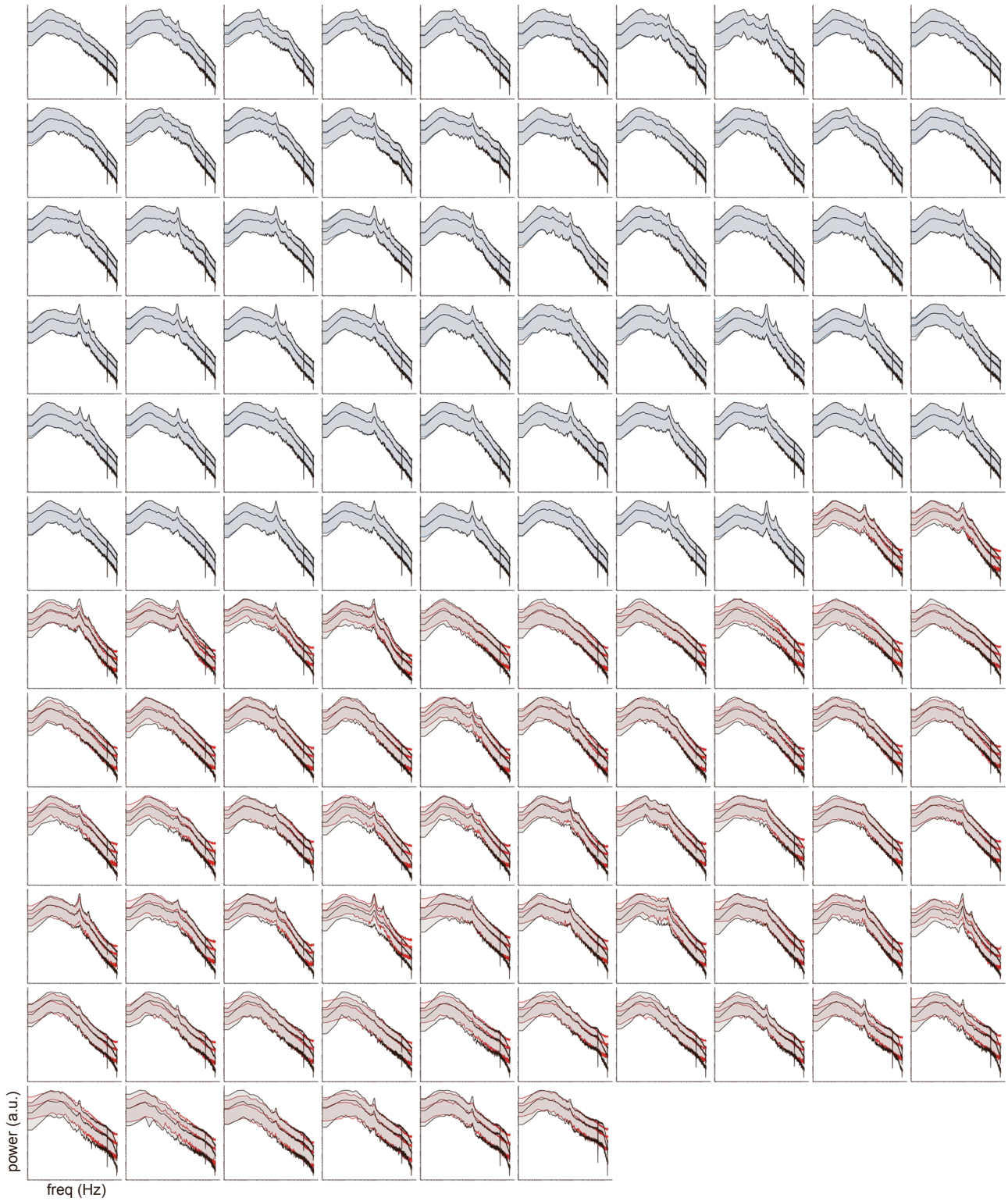


Figure S9: **Full spectra of P12 from the AJILE12 dataset.** The LFP traces consist of 126 channels. Here, we imputed the second half of the channels given the first half for all time series in the evaluation set. Median power as well as 10 %/90 % percentiles are shown.

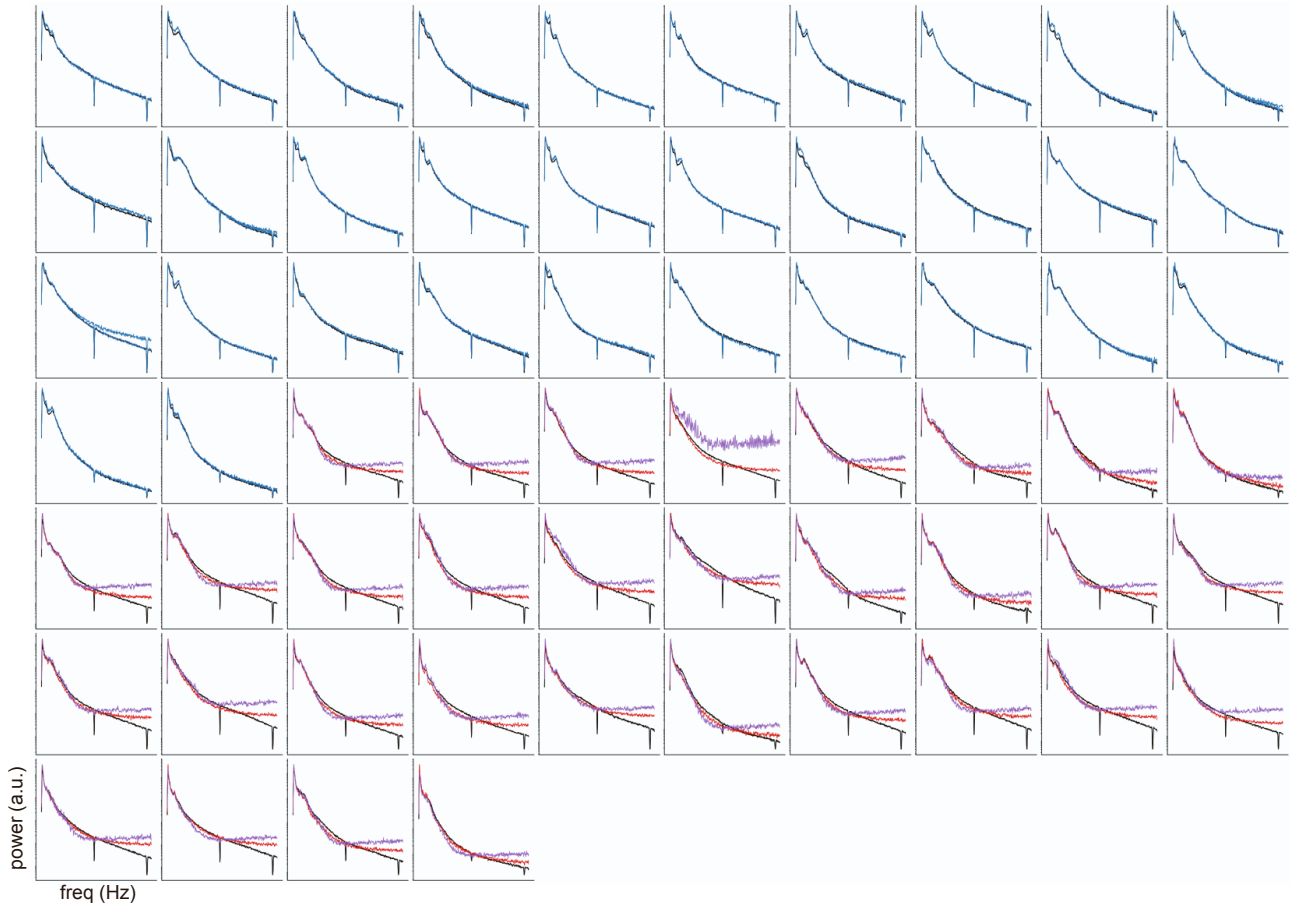


Figure S10: **OU versus white noise diffusion process for P01.** Full (semi-log) median spectra of P01 from the AJILE12 dataset. The second half of channels was imputed given the first half (blue) using a DDPM trained with either white noise (purple) or OU noise (red).

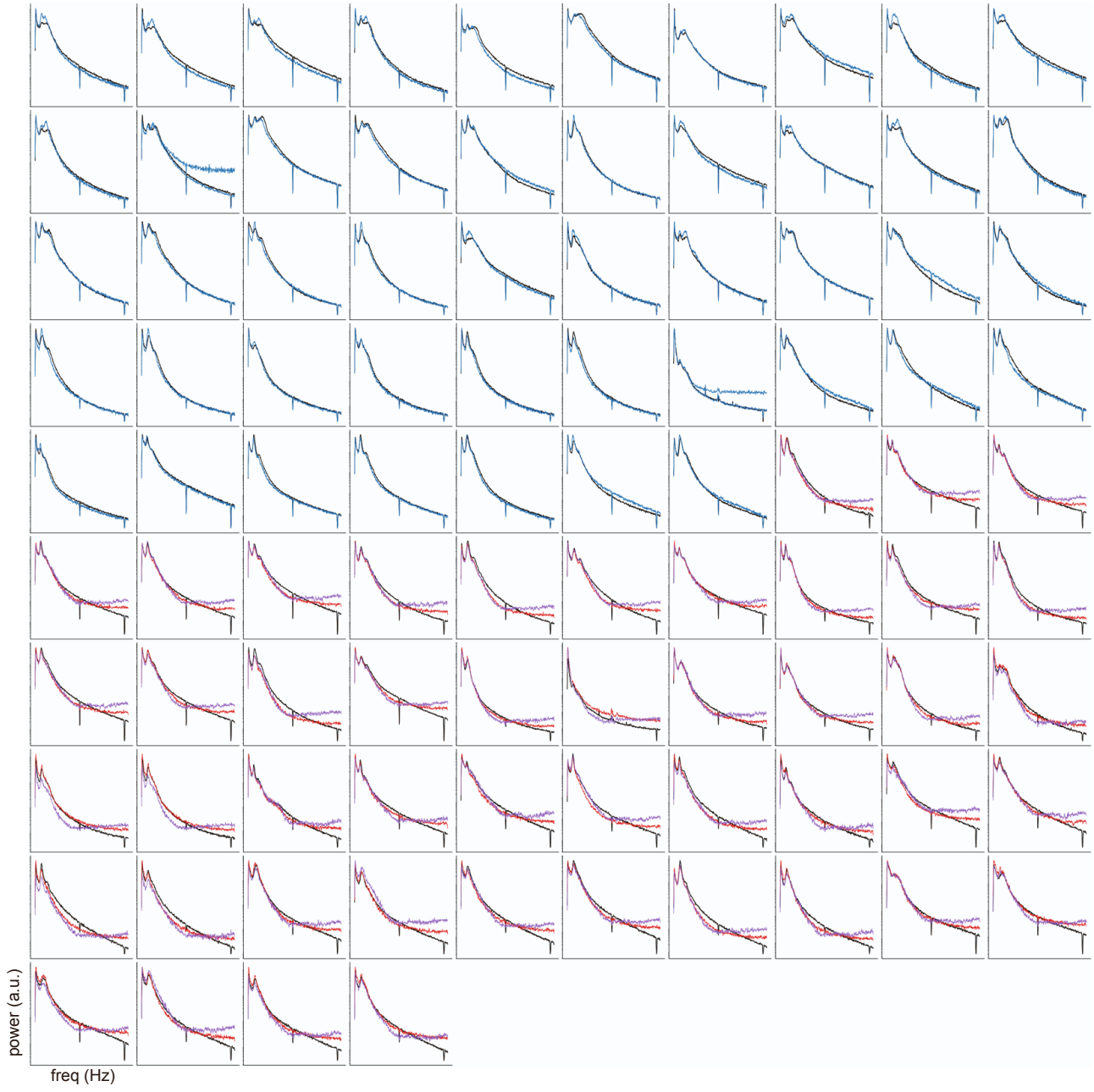


Figure S11: **OU versus white noise diffusion process for P07.** Full (semi-log) median spectra of P07 from the AJILE12 dataset. The second half of channels was imputed given the first half (blue) using a DDPM trained with either white noise (purple) or OU noise (red).

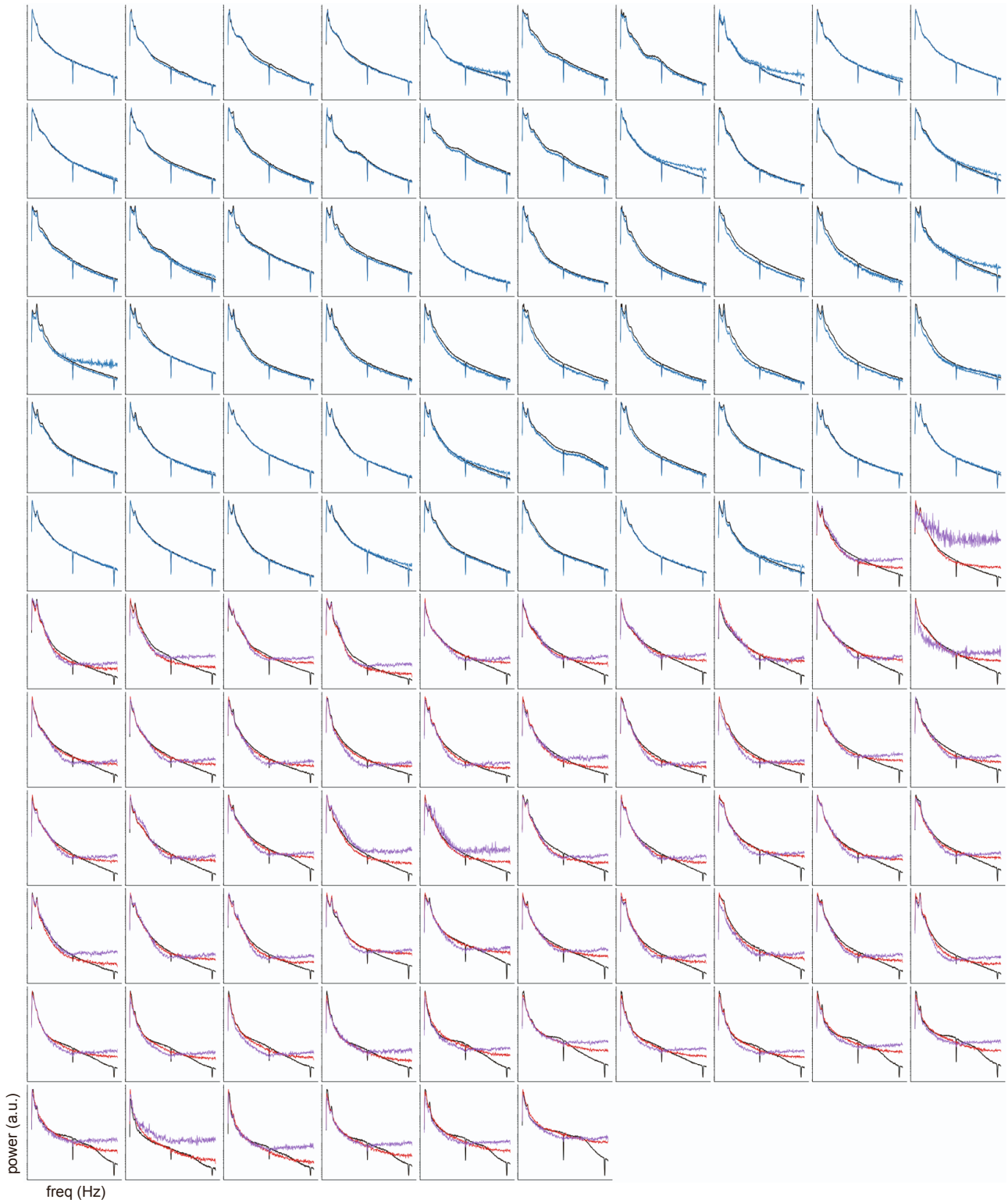


Figure S12: **OU versus white noise diffusion process for P12.** Full (semi-log) median spectra of P12 from the AJILE12 dataset. The second half of channels was imputed given the first half (blue) using a DDPM trained with either white noise (purple) or OU noise (red).

References

1. Biloš, M., Rasul, K., Schneider, A., Nevmyvaka, Y., and Günnemann, S. (2023). Modeling temporal data as continuous functions with process diffusion. International Conference on Machine Learning *202*, 2452–2470.

2. Talukder, S., Sun, J. J., Leonard, M., Brunton, B. W., and Yue, Y. (2022). Deep neural imputation: A framework for recovering incomplete brain recordings. NeurIPS 2022 Workshop on Learning from Time Series for Health.

3. Cazelles, E., Robert, A., and Tobar, F. (2020). The Wasserstein-Fourier distance for stationary time series. IEEE Transactions on Signal Processing *69*, 709–721.

4. Aznan, N. K. N., Atapour-Abarghouei, A., Bonner, S., Connolly, J. D., Al Moubayed, N., and Breckon, T. P. (2019). Simulating brain signals: Creating synthetic eeg data via neural-based generative models for improved ssvep classification. In: 2019 International joint conference on neural networks (IJCNN). (1–8).

5. Kingma, D. P., Welling, M. et al. (2019). An introduction to variational autoencoders. Foundations and Trends in Machine Learning *12*, 307–392.

6. Bredell, G., Flouris, K., Chaitanya, K., Erdil, E., and Konukoglu, E. (2023). Explicitly minimizing the blur error of variational autoencoders. International Conference on Learning Representations *abs/2304.05939*.

7. Kodali, N., Abernethy, J., Hays, J., and Kira, Z. (2017). On convergence and stability of GANs. Preprint at arXiv. https://doi.org/10.48550/arXiv.1705.07215.

8. Mescheder, L., Geiger, A., and Nowozin, S. (2018). Which training methods for GANs do actually converge? International conference on machine learning (3481–3490).

Sourcerer: Sample-based Maximum Entropy Source Distribution Estimation

Julius Vetter[†,1,2,*] **Guy Moss**[†,1,2,*]

Cornelius Schröder[1,2] **Richard Gao**[1,2]

Jakob H. Macke[1,2,3,*]

[1]Machine Learning in Science, Excellence Cluster Machine Learning, University of Tübingen
[2]Tübingen AI Center
[3]Department Empirical Inference, Max Planck Institute for Intelligent Systems
Tübingen, Germany
[†]Equal contribution.

Abstract

Scientific modeling applications often require estimating a distribution of parameters consistent with a dataset of observations—an inference task also known as source distribution estimation. This problem can be ill-posed, however, since many different source distributions might produce the same distribution of data-consistent simulations. To make a principled choice among many equally valid sources, we propose an approach which targets the maximum entropy distribution, i.e., prioritizes retaining as much uncertainty as possible. Our method is purely sample-based—leveraging the Sliced-Wasserstein distance to measure the discrepancy between the dataset and simulations—and thus suitable for simulators with intractable likelihoods. We benchmark our method on several tasks, and show that it can recover source distributions with substantially higher entropy than recent source estimation methods, without sacrificing the fidelity of the simulations. Finally, to demonstrate the utility of our approach, we infer source distributions for parameters of the Hodgkin-Huxley model from experimental datasets with hundreds of single-neuron measurements. In summary, we propose a principled method for inferring source distributions of scientific simulator parameters while retaining as much uncertainty as possible.

1 Introduction

In many scientific and engineering disciplines, mathematical and computational simulators are used to gain mechanistic insights. A common challenge is to identify parameter settings of such simulators that make their outputs compatible with a set of empirical observations. For example, by finding a distribution of parameters that, when passed through the simulator, produces a distribution of outputs that matches that of the empirical dataset of observations.

Suppose we have a stochastic simulator with input parameters θ and output x, which allows us to generate samples from the forward model $p(x|\theta)$ (which is usually intractable). We have acquired a dataset $\mathcal{D} = \{x_1, ..., x_n\}$ of observations with empirical distribution $p_o(x)$, and want to identify

[*]{firstname.secondname}@uni-tuebingen.de
Code available at https://github.com/mackelab/sourcerer

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

a distribution $q(\theta)$ over parameters that, once passed through the simulator, yields a "pushforward" distribution of simulations $q^{\#}(x) = \int p(x|\theta)q(\theta)d\theta$ that is indistinguishable from the empirical distribution. This setting is known by different names in different disciplines, for example as *unfolding* in high energy physics [10], *stochastic inverse problems* in various disciplines [7], *population of models* in electrophysiology [30] and *population inference* in gravitational wave astronomy [55]. Adopting the terminology of Vandegar et al. [58], we refer to this task as *source distribution estimation*.

A common approach to source distribution estimation is empirical Bayes [51, 15]. Empirical Bayes uses hierarchical models in which each observation is modeled as arising from different parameters $p(x_i|\theta_i)$. The hyper-parameters of the prior (and thus the source q_ϕ) are found by optimizing the marginal likelihood $p(D) = \prod_i \int p(x_i|\theta)q_\phi(\theta)d\theta$ over ϕ. Empirical Bayes has been successfully applied to a range of applications [31, 32, 55]. However, empirical Bayes is typically not applicable to models with intractable likelihoods, which is usually the case for scientific simulators. Using surrogate models for such likelihoods, empirical Bayes has been extended to increasingly more complicated parameterizations ϕ of the source distribution, including neural networks [59, 58].

A more general issue, however, is that the source distribution problem can often be ill-posed without the introduction of a hyper-prior or other regularization principles, as also noted in Vandegar et al. [58]: Distinct source distributions $q(\theta)$ can give rise to the same data distribution $q^{\#}(x)$ when pushed through the simulator $p(x|\theta)$ (Fig. 1, illustrative example in Appendix A.7).

We here propose to use the maximum entropy principle, i.e., choosing the "maximum ignorance" distribution within a class of distributions to resolve the ill-posedness of the source distribution problem [19, 24]. The maximum entropy principle formalizes the notion that a good choice for distributions should "assume less". It has been applied to specific source distribution estimation problems in scientific disciplines such as cosmology [23] and high-energy physics [10].

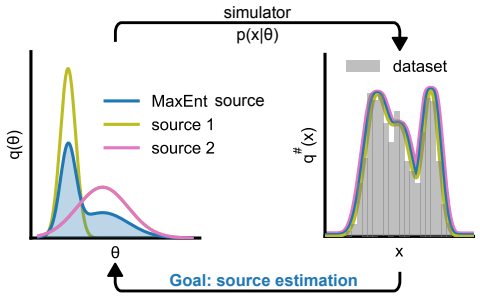


Figure 1: **Maximum entropy source distribution estimation.** Given an observed dataset $\mathcal{D} = \{x_1, \ldots, x_n\}$ from some data distribution $p_o(x)$, the *source distribution estimation* problem is to find the parameter distribution $q(\theta)$ that reproduces $p_o(x)$ when passed through the simulator $p(x|\theta)$, i.e. $q^{\#}(x) = \int p(x|\theta)q(\theta)d\theta = p_o(x)$ for all x. This problem can be ill-posed, as there might be more than one distinct source distribution. We resolve this by targeting the maximum entropy distribution, which is unique.

Our contributions We introduce *Sourcerer*, a general method for source distribution estimation, providing two key innovations: First, we target the maximum entropy source distribution to obtain a well-posed problem, thereby increasing the entropy of the estimated source distributions at no cost to their fidelity. Second, we use general distance metrics between distributions, in particular the Sliced-Wasserstein distance, instead of maximizing the marginal likelihood as in empirical Bayes. This allows evaluation of the objective using *only samples* from differentiable simulators, removing the requirement to have tractable likelihoods. We validate our method on multiple tasks, including tasks with high-dimensional observation space, which are challenging for likelihood-based methods. Finally, we apply our method to estimate the source distribution over the mechanistic parameters of the Hodgkin-Huxley model from a large ($\sim$ 1000 samples) dataset of electrophysiological recordings.

2 Methods

We formulate the source distribution estimation problem in terms of the maximum entropy principle. The (differential) entropy $H(p)$ of a distribution $p(\theta)$ is defined as

$$H(p) = -\int p(\theta) \log p(\theta) d\theta. \tag{1}$$

2.1 Data-consistency and regularized objective

For a given distribution $q(\theta)$ and a simulator with (possibly intractable) likelihood $p(x|\theta)$, the *pushforward* of q is given by $q^{\#}(x) = \int p(x|\theta)q(\theta)d\theta$. The distribution $q(\theta)$ is a source distribution if its pushfoward matches the observed data distribution $p_o(x)$, that is, $q^{\#} = p_o$ almost everywhere. Equivalently, given a distance metric $D(\cdot, \cdot)$ between probability distributions $P(\mathcal{X})$ over the data space $\mathcal{X}$, a source distribution q is one which satisfies $D(q^{\#}, p_o) = 0$. In general, for a given distribution of observations $p_o(x)$ and likelihood $p(x|\theta)$, the source distribution problem is ill-posed as there are possibly many different source distributions. The maximum entropy principle can be employed to resolve this ill-posedness:

Proposition 2.1. *Let $Q = \{q|q^{\#} = p_o\}$ be the set of source distributions for a given likelihood $p(x|\theta)$ and data distribution p_o. Suppose that Q is non-empty and compact. Then $q^* = \arg\max_{q \in Q} H(q)$ exists and is unique.*

This proposition follows from the fact that the set of source distributions is convex and that the (differential) entropy $H(q)$ is a strictly concave functional. See Appendix A.7 for a proof and additional assumptions.

Proposition 2.1 suggests to solve the constrained optimization problem

$$\max_{\phi} \quad H(q_\phi) \quad \text{s.t.} \quad D(q_\phi^{\#}, p_o) = 0, \qquad (2)$$

where q_ϕ is some parametric family of distributions.

Practically, however, a solution might not exist, for example due to simulator misspecification. Furthermore, even if a solution exists, it is difficult to obtain since we only have a fixed number of samples from p_o and can thus only estimate $D(q_\phi^{\#}, p_o)$. We therefore propose a *regularized* approximation of Eq. (2) and solve

$$\max_{\phi} \quad \lambda H(q_\phi) - (1-\lambda)\log(D(q_\phi^{\#}, p_o)) \qquad (3)$$

instead, where λ is a parameter determining the strength of the data-consistency term and the logarithm is added for numerical stability. This regularized objective is related to the Lagrangian relaxation of Eq. (2), where now $\log D(q^{\#}, p_o) \leq \log \epsilon$ for some $\epsilon > 0$ and the dual variable is $(1-\lambda)/\lambda$.

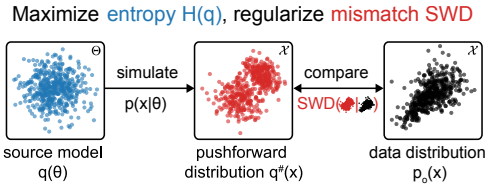


Figure 2: **Overview of Sourcerer.** Given a source distribution $q(\theta)$, we sample $\theta \sim q$ and simulate using $p(x|\theta)$ to obtain samples from the pushforward distribution $q^{\#}(x) = \int p(x|\theta)q(\theta)d\theta$. We maximize the entropy of the source distribution $q(\theta)$ while regularizing with a Sliced-Wasserstein distance (SWD) term between the pushforward of $q^{\#}$ and the data distribution $p_o(x)$ (Eq. (3)). Θ and $\mathcal{X}$ in top right corner of boxes denote parameter space and data/observation space, respectively.

For $\lambda \to 1$, the loss in Eq. (3) is dominated by the entropy term, and for $\lambda \to 0$ by the data-consistency term. We apply ideas from constrained optimization and reinforcement learning [49, 4, 1] and use a dynamical schedule during training. We initialize training with $\lambda_{t=1} = 1$, and decay this value linearly to a final value $\lambda_{t=T} = \lambda > 0$ over the course of training. This dynamical schedule encourages the variational source model to first explore high-entropy distributions, and later increase consistency with the data between high-entropy distributions. Pseudocode and details of the schedule in Appendix A.3.

2.2 Reference distribution

For many tasks, there is an additional constraint in terms of a reference distribution $p(\theta)$. For example, in the Bayesian inference framework, it is common to have a prior distribution $p(\theta)$, encoding existing knowledge about the parameters θ from previous studies. In such cases, a distribution with higher entropy than $p(\theta)$, even if it is a source distribution, is not always desirable. We therefore adapt our objective function in Eq. (3) to minimize the Kullback-Leibler (KL) divergence between the source $q(\theta)$ and the reference $p(\theta)$:

$$\min_{\phi} \quad \lambda D_{KL}(q||p) + (1-\lambda)\log(D(q^{\#}, p_o)). \qquad (4)$$

The KL divergence term can be rewritten as $D_{KL}(q||p) = -H(q) + H(q,p)$, where $H(q,p) = -\int \log(p(\theta))q(\theta)d\theta$ is the cross-entropy between q and p. Thus, provided we can evaluate the density $p(\theta)$, we can obtain a sample-based estimate of the loss in Eq. (4). In our work, we consider $p(\theta)$ to be the uniform distribution over some bounded domain B_Θ (and hence the maximum entropy distribution on this domain). This "box prior" is often used as the naive estimate from literature observations in inference studies. More specifically, in this case, $H(q,p) = -1/|B_\Theta|$, where $|B_\Theta|$ is the volume of B_Θ. Therefore, it is independent of q, and hence minimizing the KL divergence is equivalent to maximizing $H(q)$ on B_Θ. In the case where $p(\theta)$ is non-uniform (e.g., Gaussian) the cross-entropy term regularizes the loss by penalizing large $q(\theta)$ when $p(\theta)$ is small.

2.3 Sliced-Wasserstein as a distance metric

We are free to choose any distance metric $D(\cdot,\cdot)$ for the loss function Eq. (4). In this work, we use the fast, sample-based, and differentiable Sliced-Wasserstein distance (SWD) [6, 27, 42] of order two. The SWD is defined as the expected value of the one-dimensional Wasserstein distance between the projections of the distribution onto uniformly random directions u on the unit sphere $\mathbb{S}^{d-1}$ in $\mathbb{R}^d$. More precisely, the SWD is defined as

$$\text{SWD}_m(p,q) = \mathbb{E}_{u\sim\mathcal{U}(\mathbb{S}^{d-1})}[W_m(p_u, q_u)]\,, \tag{5}$$

where p_u is the one-dimensional distribution with samples $u^\top x$ for $x \sim p(x)$, and W_m is the one-dimensional Wasserstein distance of order m. In the empirical setting, where we are given n samples each from p_u and q_u respectively, the one-dimensional Wasserstein distance is computed from the order statistics as

$$W_m(p_u, q_u) = \left(\sum_{i=1}^{n} ||x_p^{(i)} - x_q^{(i)}||_m^m\right)^{1/m}, \tag{6}$$

where $x_p^{(i)}$ denotes the i-th order statistic of the samples from p_u (and similarly for $x_q^{(i)}$), and $||\cdot||_m$ denotes the L^m distance on $\mathbb{R}$ [47]. The time complexity of computing the sample-based one-dimensional Wasserstein distance is thus the time complexity of computing the order statistics, which is $\mathcal{O}(n \log n)$ in the number of datapoints n [6]. This is significantly faster than computing the multi-dimensional Wasserstein distance ($\mathcal{O}(n^3)$, 29), or the commonly used Sinkhorn algorithm for approximating the Wasserstein distance ($\mathcal{O}(n^2)$ 47). While the SWD is not the same as the multi-dimensional Wasserstein distance, it is still a valid metric on the space of probability distributions. In particular, the SWD converges quickly with rate $O(\sqrt{n})$ to its true value [41, 42].

2.4 Differentiable simulators and surrogates

Our method only requires that sampling from the simulator $p(x|\theta)$ is a differentiable operation. In practice, however, many simulators do not satisfy this property. For such simulators, we first train a surrogate model. In particular, our method can make use of surrogates that model the likelihood only implicitly. Such surrogate models can be easier to train and evaluate in practice. This is a distinct requirement from likelihood-based approaches such as Vandegar et al. [58], which require that the likelihood $p(x|\theta)$ can be evaluated explicitly *and* is differentiable. This means that our sample-based approach can be readily applied to a larger set of simulators than likelihood-based approaches.

2.5 Source model and entropy estimation

In this work we use neural samplers as proposed in Vandegar et al. [58] to parameterize a source model q_ϕ. These samplers employ unconstrained neural network architectures (in our case a multi-layer perceptron) to transform a random sample from $z \in \mathcal{N}(0, I)$ into a sample from q_ϕ. While neural samplers do not have a tractable likelihood, they are faster to evaluate than models with tractable likelihoods. Furthermore, by using unconstrained network architectures, neural samplers are flexible and additional constraints (e.g., symmetry, monotonicity) are easy to introduce.

To use likelihood-free source parameterizations, we require a purely sample-based estimator for the entropy $H(q_\phi)$. This can be done using the *Kozachenko-Leonenko* entropy estimator [28, 3], which is based on a nearest-neighbor density estimate. We use the Kozachenko-Leonenko estimator in this work for its simplicity, but note that sample-based entropy estimation is an active area of research, and other choices are possible [48]. Details about the Kozachenko-Leonenko estimator can be found in Appendix A.6.

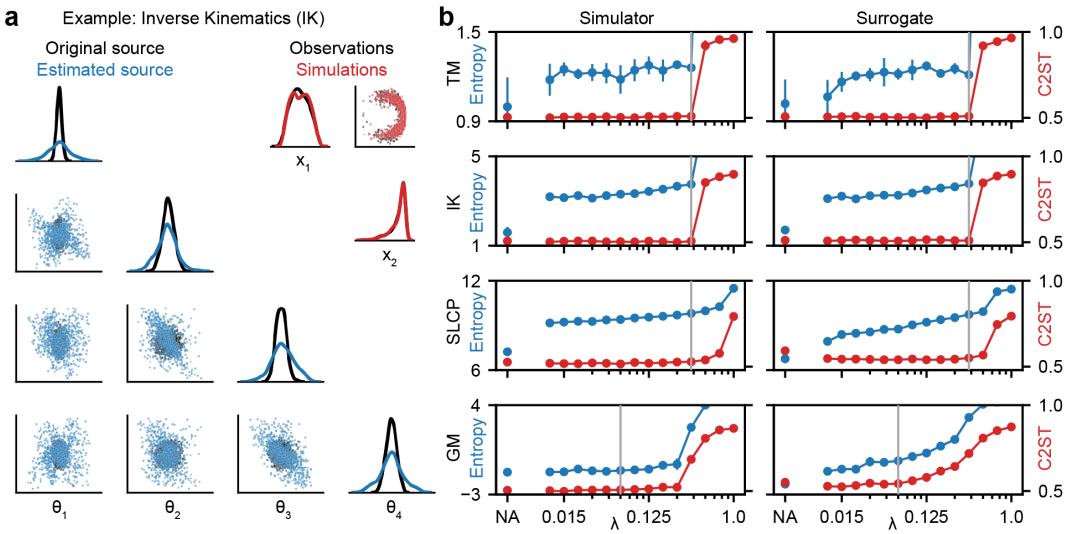


Figure 3: **Results for the source estimation benchmark. (a)** Original and estimated source and corresponding pushforward for the differentiable IK simulator ($\lambda = 0.35$). The estimated source has higher entropy than the original source that was used to generate the data. The observations (simulated with parameters from the original source) and simulations (simulated with parameters from the estimated source) match. **(b)** Performance of our approach for all four benchmark tasks (TM, IK, SLCP, GM) using both the original (differentiable) simulators, and learned surrogates. Source estimation is performed without (NA) and with entropy regularization for different choices of λ. For all cases, mean C2ST accuracy between observations and simulations (lower is better) as well as the mean entropy of estimated sources (higher is better) over five runs are shown together with the standard deviation. The gray line at $\lambda = 0.35$ ($\lambda = 0.062$ for GM) indicates our choice of final λ for the numerical benchmark results (Table 1).

3 Experiments

To evaluate the data-consistency and entropy of source distributions estimated by Sourcerer, we benchmark our method against Neural Empirical Bayes (NEB) [58], a state-of-the-art approach to source distribution estimation. The benchmark comparison is performed on four source distribution estimation tasks including three presented in Vandegar et al. [58]. We then demonstrate the advantage of Sourcerer in the case of differentiable simulators with a high-dimensional data domain, where likelihood-based empirical Bayes approaches would require training a likelihood surrogate. Finally, we use Sourcerer to estimate the source distribution for a Hodgkin-Huxley simulator of single-neuron voltage dynamics from a large dataset of experimental electrophysiological recordings. For all tasks except the Hodgkin-Huxley task (where the observed dataset is experimentally measured), we generate two datasets of observations of equal size from the same reference source distribution. The first is used to train the source model, and the second is used to evaluate the quality of the learned source.

3.1 Source Estimation Benchmark

Benchmark tasks The source estimation benchmark contains four simulators: two moons (TM), inverse kinematics (IK), simple likelihood complex posterior (SLCP), and Gaussian Mixture (GM) (details about simulators and source distributions are in Appendix A.2). Notably, all four simulators are differentiable. Therefore, we can evaluate our method directly on the simulator as well as trained surrogates. For all four simulators, source estimation is performed on a synthetic dataset of 10000 observations that were generated by sampling from a pre-defined original source distribution and evaluating the resulting pushforward distribution using the corresponding simulator. The quality of the estimated source distributions is measured using a classifier two sample test (C2ST) [33] between the observations and simulations from the source. We also report the entropy of the estimated sources. Given two sources with the same C2ST accuracy, the higher entropy source is preferable. We compare

Table 1: **Numerical benchmark results for Sourcerer.** We show the mean and standard deviation over five runs for differentiable simulators and surrogates of Sourcerer on the benchmark tasks, and compare to NEB. All approaches achieve C2ST accuracies close to 50%. For the Sliced-Wasserstein-based approach, the entropies of the estimated sources are substantially higher (bold) with the entropy regularization ($\lambda = 0.35$ for TM, IK, SLCP, $\lambda = 0.062$ for GM, gray line in Fig. 3).

Method		Sourcerer Sim. (with reg.)	Sourcerer Sim. (w/o reg.)	Sourcerer Sur. (with reg.)	Sourcerer Sur. (w/o reg.)	NEB
TM	C2ST acc.	0.51 (0.004)	0.5 (0.008)	0.51 (0.003)	0.51 (0.006)	0.53 (0.005)
	Entropy	**1.26** (0.022)	1.0 (0.198)	**1.21** (0.054)	1.02 (0.162)	1.13 (0.093)
IK	C2ST acc.	0.51 (0.002)	0.51 (0.005)	0.51 (0.005)	0.51 (0.01)	0.6 (0.014)
	Entropy	**3.75** (0.066)	1.59 (0.246)	**3.78** (0.022)	1.7 (0.165)	0.82 (0.712)
SLCP	C2ST acc.	0.53 (0.005)	0.53 (0.006)	0.55 (0.003)	0.59 (0.017)	0.53 (0.006)
	Entropy	**9.81** (0.039)	7.23 (0.052)	**9.74** (0.039)	6.76 (0.302)	7.56 (0.097)
GM	C2ST acc.	0.51 (0.005)	0.5 (0.006)	0.54 (0.006)	0.55 (0.005)	0.52 (0.004)
	Entropy	**-1.12** (0.083)	-1.25 (0.106)	**-0.36** (0.095)	-2.19 (0.212)	-1.5 (0.052)

to the NEB estimator with the same parameterization of the source model and 1024 Monte Carlo samples to estimate the marginal likelihood (details in Appendix A.3).

Benchmark performance We first check whether minimizing the Sliced-Wasserstein distance without any entropy regularization finds good source distributions. This corresponds to the case $\lambda = 0$ in Eq. (3) without any decay. In this way, we compare the data-consistency objective in Eq. (4) to the NEB objective of maximizing the marginal likelihood. We find that for the differentiable simulators, the Sliced-Wasserstein-based approach is able to find good source distributions with C2ST accuracies close to 50% for all benchmark tasks (Fig. 3, labeled NA). This also applies when we use surrogate models to generate the pushforward distributions. In particular, the quality of the estimated source distributions matches those found by NEB (Table 1).

We then apply entropy regularization as defined in Eq. (3) for all benchmark tasks. The entropy of the estimated sources is drastically increased *without* any cost in the quality of the simulations (Fig. 3b). While C2ST accuracy remains close to 50% across all benchmark tasks, the entropy of estimated sources is substantially higher than that of sources estimated with NEB, or when minimizing only the data-consistency term (Table 1). We also explore the dependence of the results on the final regularization strength λ (Fig. 3b). We observe a sharp trade-off: above a critical value of λ, the SWD term becomes too weak, and the fidelity of the simulations rapidly declines. However, below this critical value of λ, the results are robust relative to λ: the estimated sources produce simulations that match the observations, and have comparable entropy.

Additionally, for both IK and SLCP simulators, the entropy of the sources estimated by our method is higher than the entropy of the original source distribution (Fig. 3a and Fig. A7) despite the simulations and observations being indistinguishable from each other (C2ST accuracy: 50%). This does not contradict our approach: The original source distribution just happens not to be the maximum entropy source for these simulators.

We also investigate the robustness of our approach to the choice of the differentiable, sample-based distance by repeating all experiments for these benchmark tasks using the Maximum Mean Discrepancy (MMD, 22) and find comparable results (Fig. A4). Finally, we demonstrate (Fig. A5) the robustness of our approach for small dataset sizes by repeating the Two Moons task with ($N = 100$) observations (as opposed to 10000), and for high-dimensional parameter spaces by repeating the Gaussian Mixture task with $D = 25$ dimensions (as opposed to 2).

3.2 High-dimensional observations: Lotka-Volterra and SIR

Since our method is sample-based and does not require likelihoods, it is possible to estimate sources by back-propagating through the differentiable simulators directly. This is advantageous especially for simulators with high-dimensional outputs, as we no longer require to first train a surrogate likelihood model, which can be challenging when faced with high-dimensional data such as time series. Here, we

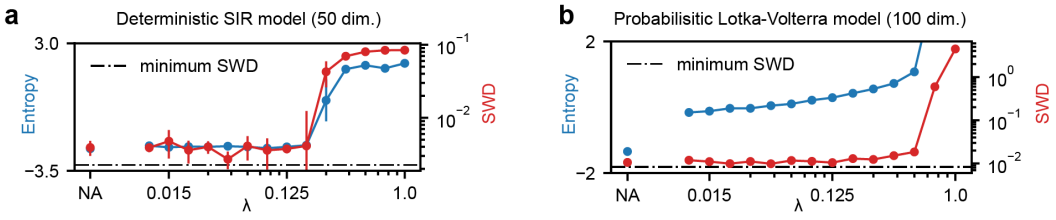


Figure 4: **Source estimation on differentiable simulators.** For both the deterministic SIR model **(a)** and probabilistic Lotka-Volterra model **(b)**, the Sliced-Wasserstein distance (lower is better) between observations and simulations as well as entropy of estimated sources (higher is better) for different choices of λ and without the entropy regularization (NA) are shown. Mean and standard deviation are computed over five runs.

highlight this capability of our method by estimating source distributions for two high-dimensional, differentiable simulators: The Lotka-Volterra model and the SIR (Susceptible, Infectious, Recovered) model. The Lotka-Volterra model is used to model the density of two populations, predators and prey. The SIR model is commonly used in epidemiology to model the spread of disease in a population (details about both models and source distributions in Appendix A.2). Compared to the benchmark tasks in Sec. 3.1, the dimensionality of the data space is much larger: Both the Lotka-Volterra and the SIR model are simulated for 50 time points resulting in a 100 and 50 dimensional time series, respectively.

Furthermore, to show that unlike NEB (which maximizes the marginal likelihood), our sample-based approach is applicable to deterministic simulators, we use a deterministic version of the SIR model with no observation noise. Similarly to the benchmark tasks, we define a source, and simulate 10000 observations using samples from this source to define a synthetic dataset on which to perform source distribution estimation. Here, we directly evaluate the quality of the estimated source distributions using the Sliced-Wasserstein distance. We compare this distance to the minimum expected distance, which is the distance between simulations of different sets of samples from the same original source. For a comparison with NEB, we train surrogate models with a reduced dimensionality and again compute C2ST accuracies and entropies of the estimated sources (see Appendix A.5 and Fig. A3 for details on surrogate training and pushforward plots).

Source estimation for the deterministic SIR model Our method is able to estimate a good source distribution for the deterministic SIR model: The Sliced-Wasserstein distance between simulations and observations is close to the minimum expected distance (Fig. 4a). In contrast to the benchmark tasks, estimating sources with entropy regularization does not lead to an increase in entropy for the SIR model, and the quality of the estimated source remains constant for various choices of λ. A possible explanation for this is that there is no degeneracy in the parameter space of the deterministic simulator, and there exists only one source distribution.

Source estimation for the probabilistic Lotka-Volterra model For the probabilistic Lotka-Volterra model, our method is also capable of estimating source distributions. As for the SIR model, the Sliced-Wasserstein distance between simulations and observations is close to the minimum expected distance (Fig. 4b). However, unlike the SIR model, estimating the source with entropy regularization yields a large increase in entropy compared to when not using the regularization. For the Lotka-Volterra model, our method yields a substantially higher entropy at no additional cost in terms of source quality.

When using the surrogate models with reduced dimensionality to estimate the source distributions, we find that Sourcerer achieves better C2ST accuracies than NEB. Furthermore, for the Lotka-Volterra model, the entropy regularization again leads to a substantial increase in the entropy of the estimated sources (Table 2). In summary, the experiments on the SIR and Lotka-Volterra models show that our approach is able to scale to higher dimensional problems and can use gradients of complex simulators to estimate source distributions directly from a set of observations.

Table 2: **Numerical results for the SIR and Lotka-Volterra model** We show the mean and standard deviation over five runs for differentiable simulators and surrogates of Sourcerer on the high-dimensional SIR and Lotka-Volterra (LV) models, and compare to NEB. For the comparison with NEB, we train the required surrogate models with reduced dimensionality (25 dimensions instead of 50 or 100). Sourcerer achieves C2ST accuracies close to 50%. For NEB, the C2ST accuracies are worse. For the LV model, the entropies of the estimated sources are higher with the entropy regularization ($\lambda = 0.015$ for SIR, $\lambda = 0.125$ for LV).

Method		Sourcerer Sim. (with reg.)	Sourcerer Sim. (w/o reg.)	Sourcerer Sur. (with reg.)	Sourcerer Sur. (w/o reg.)	NEB
SIR	C2ST acc.	0.56 (0.013)	0.56 (0.015)	0.55 (0.005)	0.55 (0.005)	0.76 (0.024)
	Entropy	-2.3 (0.079)	-2.37 (0.169)	-2.29 (0.076)	-2.5 (0.05)	-0.63 (0.174)
LV	C2ST acc.	0.57 (0.009)	0.52 (0.001)	0.56 (0.005)	0.54 (0.009)	0.62 (0.011)
	Entropy	**0.29** (0.017)	-1.34 (0.087)	**0.34** (0.05)	-1.01 (0.13)	-1.28 (0.073)

3.3 Estimating source distributions for a single-compartment Hodgkin-Huxley model

Single-compartment Hodgkin-Huxley simulator and summary statistics The single-compartment Hodgkin-Huxley model consists of a system of coupled ordinary differential equations simulating different ion channels in a neuron. We use the simulator described in Bernaerts et al. [2] with 13 parameters. In data space, we use five commonly used summary statistics of the observed and simulated spike trains. These are the (log of the) number of spikes, the mean of the resting potential, and the mean, variance and skewness of the voltage during external current stimulation. As the internal noise in the simulator has little effect on the summary statistics, we train a simple multi-layer perceptron as surrogate on 10^6 simulations. The parameters used to generate these training simulations were sampled from a uniform distribution that was used as the prior in Bernaerts et al. [2] (details on simulator, choice of surrogate and the surrogate training in Appendix A.9).

Using this surrogate, we estimate source distributions from a real-world dataset of electrophysiological recordings. The dataset [52] consists of 1033 electrophysiological recordings from the mouse motor cortex. In general, parameter inference for Hodgkin-Huxley models can be challenging as models are often misspecified [56, 2]. Thus, estimating the source distribution for this task is useful for downstream inference tasks, as the prior knowledge gained can significantly constrain the parameters of interest.

Source estimation for the Hodgkin-Huxley model On visual inspection, simulations from the estimated source look similar to the original recordings (all observations spike at least once, spikes have similar magnitudes) and show none of the unrealistic properties (e.g., spiking before the stimulus is applied) that can be observed in some of the box uniform prior simulations (Fig. 5a). This match is also confirmed by the distribution of summary statistics, which match closely between simulations and observations (Fig. 5b). Furthermore, our method achieves good C2ST accuracy of $\approx 61\%$ for different choices of λ (Fig. 5d), as well as a small Sliced-Wasserstein distance of ≈ 0.08 in the standardized space of summary statistics (Fig. 5e). While the source estimated without entropy regularization also achieves good fidelity, its entropy is significantly lower than any of the source distributions estimated with entropy regularization (Fig. 5d/e, example source distribution in Fig. 5c, full source in Fig. A11).

Overall, these results demonstrate the importance of estimating source distributions using the entropy regularization, especially on real-world datasets: Estimating the source distribution without any entropy regularization can introduce severe bias, since the estimated source may ignore entire regions of the parameter space. In this example, the parameter space of the single-compartment Hodgkin-Huxley model is known to be highly degenerate, and a given observation can be generated by multiple parameter configurations [14, 39].

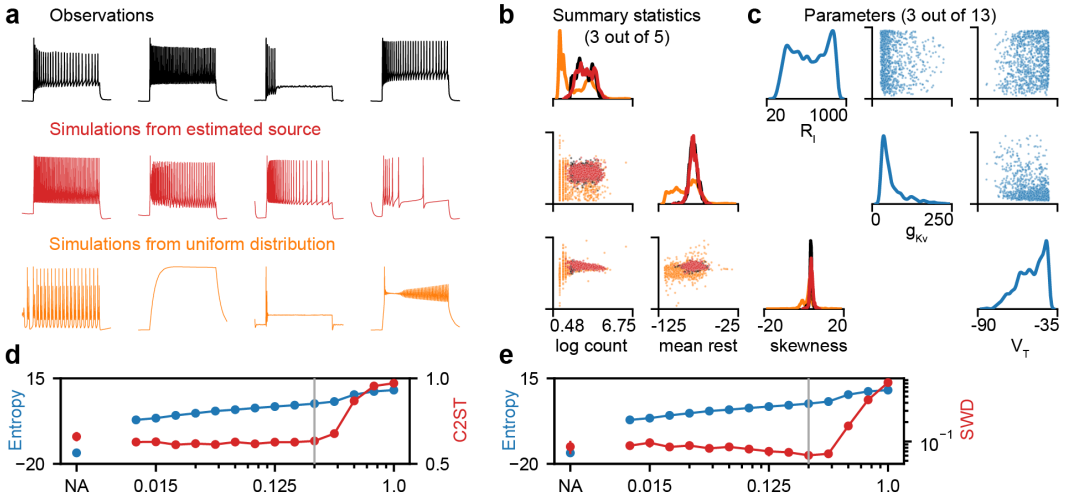


Figure 5: **Source estimation for the single-compartment Hodgkin-Huxley model. (a)** Example voltage traces of the real observations of the motor cortex dataset, simulations from the estimated source ($\lambda = 0.25$), and samples from the uniform distribution used to train the surrogate. **(b)** 1D and 2D marginals for three of the five summary statistics used to perform source estimation. **(c)** 1D and 2D marginal distributions of the estimated source for three of the 13 simulator parameters. **(d)** and **(e)** C2ST accuracy and Sliced-Wasserstein distance (lower is better) as well as entropy of estimated sources (higher is better) for different choices of λ including $\lambda = 0.25$ (gray line) and without entropy regularization (NA). Mean and standard deviation over five runs are shown.

4 Related Work

Neural Empirical Bayes High-dimensional source distributions have been estimated through variational approximations to the empirical Bayes problem. Louppe et al. [34] train a generative adversarial network (GAN) [20] q_ψ to approximate the source. The use of a discriminator to compute an implicit distance makes this approach purely sample-based as well. In order to find the optimal ψ^* of the true data-generating process, they augment the adversarial loss with a small entropy penalty on the source q_ψ. This penalty encourages low entropy, point mass distributions, which is the *opposite* of our approach. Vandegar et al. [58] take an empirical Bayes approach, and use normalizing flows for both the variational approximation of the source and as a surrogate for the likelihood $p(x|\theta)$. This allows for direct regression on the marginal likelihood, as all likelihoods can be computed directly. Finally, the empirical Bayes problem is also known as "unfolding" in the particle physics literature [10], "population inference" in gravitational wave astronomy [55], and "population of models" in electrophysiology [30]. Approaches have been developed to identify the source distribution, including classical approaches that seek to increase the entropy of the learned sources [50].

Simulation-Based Inference The use of variational surrogates of the likelihood of a simulator with intractable likelihood is known as *Neural Likelihood Estimation* in the simulation-based inference (SBI) literature [60, 45, 36, 11]. In neural posterior estimation [44, 35, 21], an *amortized* posterior density estimate is learned, which can be applied to evaluate the posterior of a single observation $x_i \in \mathcal{D}$, if a prior distribution $p(\theta)$ is already known. An intuitive but incorrect approach to source distribution estimation would be to take the *average posterior* distribution over the observations $\mathcal{D}$,

$$G_n(\theta) = \frac{1}{n} \sum_{i=1}^{n} p(\theta|x_i). \tag{7}$$

The average posterior does not always (and typically does not) converge to a source distribution in the infinite data limit, as shown for simple examples in Appendix A.8. Intuitively, the average posterior becomes a worse approximation of a source distribution for simulators that have broader likelihoods. Instead, SBI can be seen as a downstream task of source distribution estimation; once a prior has been learned from the dataset of observations with source estimation, the posterior can be estimated for each new observation individually.

Generalized Bayesian Inference Another field related to source estimation is Generalized Bayesian Inference (GBI) [5, 40, 26]. GBI performs distance-based inference, as opposed to targeting the exact Bayesian posterior. Similarly to our work, the distance function used in GBI can be arbitrarily chosen for different tasks. However, GBI is used for single-parameter inference tasks, as opposed to the source distribution estimation task considered in this work. Similarly, Bayesian non-parametric methods [43, 38, 12] learn a posterior directly on the data space which can then be used to sample from a posterior distribution over the parameter space.

5 Summary and Discussion

In this work, we introduced Sourcerer as a method to estimate source distributions of simulator parameters given datasets of observations. This is a common problem setting across a range of scientific and engineering disciplines. Our method has several advantages: first, we employ a maximum entropy approach, improving reproducibility of the learned source, as the maximum entropy source distribution is unique while the traditional source distribution estimation problem can be ill-posed. Second, our method allows for sample-based optimization. In contrast to previous likelihood-based approaches, this scales more readily to higher dimensional problems, and can be applied to simulators without a tractable likelihood. We demonstrated the performance of our approach across a diverse suite of tasks, including deterministic and probabilistic simulators, differentiable simulators and surrogate models, low- and high-dimensional observation spaces, and a contemporary scientific task of estimating a source distribution for the single-compartment Hodgkin-Huxley model from a dataset of electrophysiological recordings. Throughout our experiments, we have consistently found that our approach yields higher entropy sources without reducing the fidelity of simulations from the learned source.

Limitations In this work, we used the Sliced-Wasserstein distance (and MMD) for the data-consistency term between simulations and observations. In practice, different distance metrics can lead to different estimated sources, depending on its sensitivity to different features. While our method is compatible with any sample-based differentiable distance metric between two distributions, there is still an onus on the practitioner to carefully select a reasonable distance metric for the data at hand. For example, in some cases, it might be appropriate to use a combination of several distance metrics for different modalities of the data. Similarly, there is a dependence on the final regularization strength λ. Principled methods for defining the regularization strength are desirable, though as we demonstrate, our results are robust to a large range of λ.

In addition, the method requires a differentiable simulator, which in practice may require the training of a surrogate model, for example, when dealing with a (partially) discrete simulator. While this is a common requirement for simulation-based methods, this could present a challenge for some applications. Finally, in our work, we enforce the maximum entropy principle on the entire (parameter) source distribution. In practice, for example when constructing prior distributions for Bayesian inference, there are other choices, such as the Jeffrey's prior [9].

Acknowledgements

This work was funded by the German Research Foundation (DFG) under Germany's Excellence Strategy – EXC number 2064/1 – 390727645 and SFB 1233 'Robust Vision' (276693517). This work was co-funded by the German Federal Ministry of Education and Research (BMBF): Tübingen AI Center, FKZ: 01IS18039A and the European Union (ERC, DeepCoMechTome, 101089288). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them. JV is supported by the AI4Med-BW graduate program. JV and GM are members of the International Max Planck Research School for Intelligent Systems (IMPRS-IS). We would like to thank Jonas Beck, Sebastian Bischoff, Michael Deistler, Manuel Glöckler, Jaivardhan Kapoor, Auguste Schulz, and all members of Mackelab for feedback and discussion throughout the project.

References

[1] Zafarali Ahmed, Nicolas Le Roux, Mohammad Norouzi, and Dale Schuurmans. Understanding the impact of entropy on policy optimization. In *International conference on machine learning*, 2019.

[2] Yves Bernaerts, Michael Deistler, Pedro J Goncalves, Jonas Beck, Marcel Stimberg, Federico Scala, Andreas S Tolias, Jakob H Macke, Dmitry Kobak, and Philipp Berens. Combined statistical-mechanistic modeling links ion channel genes to physiology of cortical neuron types. *bioRxiv*, 2023.

[3] Thomas B. Berrett, Richard J. Samworth, and Ming Yuan. Efficient multivariate entropy estimation via k-nearest neighbour distances. *The Annals of Statistics*, 2019.

[4] D.P. Bertsekas and W. Rheinboldt. *Constrained Optimization and Lagrange Multiplier Methods*. Computer science and applied mathematics. Elsevier Science, 2014.

[5] Pier Giovanni Bissiri, Chris C Holmes, and Stephen G Walker. A general framework for updating belief distributions. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 2016.

[6] Nicolas Bonneel, Julien Rabin, Gabriel Peyré, and Hanspeter Pfister. Sliced and Radon Wasserstein barycenters of measures. *Journal of Mathematical Imaging and Vision*, 2015.

[7] T. Butler, J. Jakeman, and T. Wildey. Combining push-forward measures and bayes' rule to construct consistent solutions to stochastic inverse problems. *SIAM Journal on Scientific Computing*, 2018.

[8] E.K.P. Chong, W.S. Lu, and S.H. Zak. *An Introduction to Optimization: With Applications to Machine Learning*. Wiley, 2023.

[9] Guido Consonni, Dimitris Fouskakis, Brunero Liseo, and Ioannis Ntzoufras. Prior Distributions for Objective Bayesian Analysis. *Bayesian Analysis*, 2018.

[10] G. Cowan. *Statistical Data Analysis*. Oxford science publications. Clarendon Press, 1998.

[11] Kyle Cranmer, Johann Brehmer, and Gilles Louppe. The frontier of simulation-based inference. *Proceedings of the National Academy of Sciences*, 2019.

[12] Charita Dellaporta, Jeremias Knoblauch, Theodoros Damoulas, and François-Xavier Briol. Robust Bayesian inference for simulator-based models via the MMD posterior bootstrap. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 2022.

[13] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using Real NVP. In *International Conference on Learning Representations*, 2017.

[14] Gerald M Edelman and Joseph A Gally. Degeneracy and complexity in biological systems. *Proceedings of the National Academy of Sciences*, 2001.

[15] Bradley Efron and Carl Morris. Limiting the risk of Bayes and empirical Bayes estimators, part ii: The empirical Bayes case. *Journal of the American Statistical Association*, 1972.

[16] Philip E. Gill, Walter Murray, and Margaret H. Wright. *Practical Optimization*. Society for Industrial and Applied Mathematics, 2019.

[17] Manuel Glöckler, Michael Deistler, and Jakob H. Macke. Adversarial robustness of amortized Bayesian inference. In *International Conference on Machine Learning*, 2023.

[18] 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*, 2020.

[19] I. J. Good. Maximum entropy for hypothesis formulation, especially for multidimensional contingency tables. *Annals of Mathematical Statistics*, 1963.

[20] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, 2014.

[21] David S. Greenberg, Marcel Nonnenmacher, and Jakob H. Macke. Automatic posterior transformation for likelihood-free inference. In *International Conference on Machine Learning*, 2019.

[22] A Gretton, KM. Borgwardt, MJ. Rasch, B Schölkopf, and Alexander Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 2012.

[23] Will Handley and Marius Millea. Maximum-entropy priors with derived parameters in a specified distribution. *Entropy*, 2018.

[24] Edwin T. Jaynes. Prior probabilities. *IEEE Transactions on Systems Science and Cybernetics*, 1968.

[25] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[26] Jeremias Knoblauch, Jack Jewson, and Theodoros Damoulas. An optimization-centric view on bayes' rule: Reviewing and generalizing variational inference. *Journal of Machine Learning Research*, 2022.

[27] Soheil Kolouri, Kimia Nadjahi, Umut Simsekli, Roland Badeau, and Gustavo Rohde. Generalized Sliced Wasserstein distances. In *Advances in Neural Information Processing Systems*, 2019.

[28] L. Kozachenko and N. Leonenko. A statistical estimate for the entropy of a random vector. *Problems of Information Transmission*, 1987.

[29] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 1955.

[30] Brodie A. J. Lawson, Christopher C. Drovandi, Nicole Cusimano, Pamela Burrage, Blanca Rodriguez, and Kevin Burrage. Unlocking data sets by calibrating populations of models to data density: A study in atrial electrophysiology. *Science Advances*, 2018.

[31] Tai Sing Lee and David Mumford. Hierarchical Bayesian inference in the visual cortex. *J. Opt. Soc. Am. A*, 2003.

[32] Ning Leng, John A. Dawson, James A. Thomson, Victor Ruotti, Anna I. Rissman, Bart M. G. Smits, Jill D. Haag, Michael N. Gould, Ron M. Stewart, and Christina Kendziorski. EBSeq: an empirical bayes hierarchical model for inference in RNA-seq experiments. *Bioinformatics*, 2013.

[33] David Lopez-Paz and Maxime Oquab. Revisiting classifier two-sample tests. In *International Conference on Learning Representations*, 2017.

[34] Gilles Louppe, Joeri Hermans, and Kyle Cranmer. Adversarial variational optimization of non-differentiable simulators. In *International Conference on Artificial Intelligence and Statistics*, 2019.

[35] 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. In *Advances in Neural Information Processing Systems*, 2017.

[36] Jan-Matthis Lueckmann, Giacomo Bassetto, Theofanis Karaletsos, and Jakob H. Macke. Likelihood-free inference with emulator networks. In *Proceedings of The 1st Symposium on Advances in Approximate Bayesian Inference*, 2019.

[37] Jan-Matthis Lueckmann, Jan Boelts, David Greenberg, Pedro Goncalves, and Jakob Macke. Benchmarking simulation-based inference. In *International Conference on Artificial Intelligence and Statistics*, 2021.

[38] Simon Lyddon, Chris C. Holmes, and Stephen G. Walker. General Bayesian updating and the loss-likelihood bootstrap. *Biometrika*, 2017.

[39] Eve Marder and Adam L Taylor. Multiple models to capture the variability in biological neurons and networks. *Nature neuroscience*, 2011.

[40] Takuo Matsubara, Jeremias Knoblauch, François-Xavier Briol, and Chris J Oates. Robust generalised Bayesian inference for intractable likelihoods. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 2022.

[41] Kimia Nadjahi, Alain Durmus, Umut Simsekli, and Roland Badeau. Asymptotic guarantees for learning generative models with the sliced-wasserstein distance. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2019.

[42] Kimia Nadjahi, Alain Durmus, Lénaïc Chizat, Soheil Kolouri, Shahin Shahrampour, and Umut Simsekli. Statistical and topological properties of sliced probability divergences. In *Advances in Neural Information Processing Systems*, 2020.

[43] Peter Orbanz and Yee Whye Teh. Bayesian nonparametric models. *Encyclopedia of Machine Learning*, 2010.

[44] George Papamakarios and Iain Murray. Fast ϵ-free inference of simulation models with Bayesian conditional density estimation. In *Advances in Neural Information Processing Systems*, 2016.

[45] George Papamakarios, David C. Sterratt, and Iain Murray. Sequential neural likelihood: Fast likelihood-free inference with autoregressive flows. In *International Conference on Artificial Intelligence and Statistics*, 2018.

[46] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[47] Gabriel Peyré and Marco Cuturi. Computational optimal transport. *Found. Trends Mach. Learn.*, 2018.

[48] Georg Pichler, Pierre Colombo, Malik Boudiaf, Günther Koliander, and Pablo Piantanida. A differential entropy estimator for training neural networks. In *International Conference on Machine Learning*, 2022.

[49] John Platt and Alan Barr. Constrained differential optimization. In *Neural Information Processing Systems*, 1987.

[50] Marcel Reginatto, Paul Goldhagen, and Sonja Neumann. Spectrum unfolding, sensitivity analysis and propagation of uncertainties with the maximum entropy deconvolution code MAXED. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 2002.

[51] Herbert E. Robbins. An empirical bayes approach to statistics. In *Breakthroughs in Statistics: Foundations and basic theory*, 1956.

[52] Federico Scala, Dmitry Kobak, Matteo Bernabucci, Yves Bernaerts, Cathryn René Cadwell, Jesus Ramon Castro, Leonard Hartmanis, Xiaolong Jiang, Sophie Laturnus, Elanine Miranda, et al. Phenotypic variation of transcriptomic cell types in mouse motor cortex. *Nature*, 2021.

[53] Scott A Sisson, Yanan Fan, and Mark M Tanaka. Sequential monte carlo without likelihoods. *Proceedings of the National Academy of Sciences*, 2007.

[54] 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. *Journal of Open Source Software*, 2020.

[55] Eric Thrane and Colm Talbot. An introduction to Bayesian inference in gravitational-wave astronomy: Parameter estimation, model selection, and hierarchical models. *Publications of the Astronomical Society of Australia*, 2019.

[56] Nicholas Tolley, Pedro LC Rodrigues, Alexandre Gramfort, and Stephanie Jones. Methods and considerations for estimating parameters in biophysically detailed neural models with simulation based inference. *bioRxiv*, 2023.

[57] Pravin M. Vaidya. An O(n log n) algorithm for the all-nearest-neighbors problem. *Discrete & Computational Geometry*, 1989.

[58] Maxime Vandegar, Michael Kagan, Antoine Wehenkel, and Gilles Louppe. Neural empirical Bayes: Source distribution estimation and its applications to simulation-based inference. In *International Conference on Artificial Intelligence and Statistics*, 2020.

[59] Yixin Wang, Andrew C. Miller, and David M. Blei. Comment: Variational Autoencoders as Empirical Bayes. *Statistical Science*, 2019.

[60] Simon N. Wood. Statistical inference for noisy nonlinear ecological dynamic systems. *Nature*, 2010.

[61] Omry Yadan. Hydra - a framework for elegantly configuring complex applications. Github, 2019. URL https://github.com/facebookresearch/hydra.

A Appendix

A.1 Software and data

We use PyTorch [46] for the source distribution estimation and hydra [61] to track all configurations. Code to reproduce results is available at https://github.com/mackelab/sourcerer.

A.2 Simulators and sources

Here we provide a definition of the four benchmark tasks Two Moons (TM), Inverse Kinematics (IK), Simple Likelihood Complex Posterior (SLCP) and Gaussian Mixture (GM), as well as the two high-dimensional simulators, the SIR and Lotka-Volterra model. We also describe the original source distribution used to generate the synthetic observations, and the bounds of the reference uniform distribution on the parameters.

A.2.1 Two moons simulator

Dimensionality	$x \in \mathbb{R}^2, \theta \in \mathbb{R}^2$
Bounded domain	$[-5, 5]^2$
Original source	$\theta \sim \mathcal{U}([-1, 1]^2)$
Simulator	$\boldsymbol{x}\|\theta = \begin{bmatrix} r\cos(\alpha) + 0.25 \\ r\sin(\alpha) \end{bmatrix} + \begin{bmatrix} -\|\theta_1 + \theta_2\|/\sqrt{2} \\ (-\theta_1 + \theta_2)/\sqrt{2} \end{bmatrix}$, where $\alpha \sim U(-\pi/2, \pi/2)$, $r \sim \mathcal{N}(0.1, 0.01^2)$.
References	Vandegar et al. [58], Lueckmann et al. [37]

A.2.2 Inverse Kinematics simulator

Dimensionality	$x \in \mathbb{R}^2, \theta \in \mathbb{R}^4$
Bounded domain	$[-\pi, \pi]^4$
Original source	$\theta \sim \mathcal{N}(0, \text{Diag}(\frac{1}{2}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}))$
Simulator	$x_1 = \theta_1 + l_1 \sin(\theta_2 + \epsilon) + l_2 \sin(\theta_2 + \theta_3 + \epsilon) + l_3 \sin(\theta_2 + \theta_3 + \theta_4 + \epsilon)$, $x_2 = l_1 \cos(\theta_2 + \epsilon) + l_2 \cos(\theta_2 + \theta_3 + \epsilon) + l_3 \cos(\theta_2 + \theta_3 + \theta_4 + \epsilon)$, where $l_1 = l_2 = 0.5$, $l_3 = 1.0$ and $\epsilon \sim \mathcal{N}(0, 0.00017^2)$.
References	Vandegar et al. [58]

A.2.3 SLCP simulator

Dimensionality	$x \in \mathbb{R}^8, \theta \in \mathbb{R}^5$
Bounded domain	$[-5, 5]^5$
Original source	$\theta \sim \mathcal{U}([-3, 3]^5)$
Simulator	$x\|\theta = (x_1, \ldots, x_4)$, $x_i \sim \mathcal{N}(m_\theta, S_\theta)$, where $m_\theta = \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix}$, $S_\theta = \begin{bmatrix} s_1^2 & \rho s_1 s_2 \\ \rho s_1 s_2 & s_2^2 \end{bmatrix}$, $s_1 = \theta_3^2, s_2 = \theta_4^2, \rho = \tanh\theta_5$.
References	Vandegar et al. [58], Lueckmann et al. [37]

A.2.4 Gaussian mixture simulator

Dimensionality	$x \in \mathbb{R}^2, \theta \in \mathbb{R}^2$
Bounded domain	$[-5, 5]^2$
Original source	$\theta \sim \mathcal{U}([0.5, 1]^2)$
Simulator	$x\|\theta \sim 0.5\mathcal{N}(x\|\theta, I) + 0.5\mathcal{N}(x\|\theta, 0.01 \cdot I)$.
References	Sisson et al. [53]

A.2.5 SIR model

Dimensionality	$x \in \mathbb{R}^{50}$, $\theta \in \mathbb{R}^2$
Bounded domain	$[0.001, 3]^2$
Original source	$\beta \sim LogNormal(\log(0.4), 0.5)$ $\gamma \sim LogNormal(\log(0.125), 0.2)$
Simulator	$x\|\theta = (x_1, \dots, x_{50})$, where $x_i = I_i/N$ equally spaced and I is simulated from $\frac{dS}{dt} = -\beta\frac{SI}{N}$, $\frac{dI}{dt} = \beta\frac{SI}{N} - \gamma I$, $\frac{dR}{dt} = \gamma I$ with initial values $S = N - 1$, $I = 1$, $R = 0$ and $N = 10^6$.
References	Lueckmann et al. [37]

A.2.6 Lotka-Volterra model

Dimensionality	$x \in \mathbb{R}^{100}$, $\theta \in \mathbb{R}^4$
Bounded domain	$[0.1, 3]^4$
Original source	$\theta' \sim \mathcal{N}(0, 0.5^2)^4$, pushed through $\theta = f(\theta') = \exp(\sigma(\theta'))$, where σ is the sigmoid function.
Simulator	$x\|\theta = (x_1^X, \dots, x_{50}^X, x_1^Y, \dots, x_{50}^Y)$, where $x_i^X \sim \mathcal{N}(X, 0.05^2)$, $x_i^Y \sim \mathcal{N}(Y, 0.05^2)$ equally spaced, and X, Y are simulated from $\frac{dX}{dt} = \alpha X - \beta XY$, $\frac{dY}{dt} = -\gamma Y + \delta XY$ with initial values $X = Y = 1$.
References	Glöckler et al. [17]

A.3 Pseudocode and details on source estimation for benchmark tasks

Pseudocode for Sourcerer is provided in Algorithm 1.

For both the benchmark tasks and high dimensional simulators, sources were estimated from 10000 synthetic observations that were generated by simulating samples from an original previously defined source.

For the benchmark tasks, we used $T = 500$ linear decay steps from $\lambda_{t=0}$ to $\lambda_{t=T} = \lambda$ and optimized the source model using the Adam optimizer with a learning rate of 10^{-4} and weight decay of 10^{-5}. The two high dimensional simulators were optimized with a higher learning rate of 10^{-3} and $T = 50$ linear decay steps. In both cases, early stopping was performed when the overall loss in Eq. (4) did not improve over a set number of training iterations.

As a baseline, we compare to Neural Empirical Bayes (NEB) as described in Vandegar et al. [58]. Specifically, we use the biased estimator with 1024 samples per observation ($\mathcal{L}_{1024}$), which are used to compute the Monte Carlo integral. Unlike our Sliced-Wasserstein-based approach, NEB does not operate on the whole dataset of observations directly but attempts to maximize the marginal likelihood per observation and thus uses part of the observations as a validation set. To ensure a fair comparison, we increased the number of observations to 11112 for all NEB experiments, which results in a training dataset of 10000 observations when using 10% as a validation set. For training, we again used the Adam optimizer (learning rate 10^{-4}, weight decay 10^{-5}, training batch size 128).

A.4 Source model

Throughout all our experiments, we use neural samplers as the source models [58]. The sampler architecture is a three-layer multi-layer perceptron with dimension of 100, ReLU activations and batch normalization as our source model. Samples are generated by drawing a sample $s \sim \mathcal{N}(0, I)$ from the standard multivariate Gaussian and then (non-linearly) transforming s with the neural network.

A.5 Surrogates for the benchmark tasks

We follow Vandegar et al. [58] and train RealNVP flows [13] as surrogates for the four benchmark tasks. For all benchmark tasks, the RealNVP surrogates have a flow length of 8 layers with a hidden dimension of 50.

Surrogates for the benchmark tasks were trained using the Adam optimizer [25] on 15000 samples and simulator evaluations from the uniform distribution over the bounded domain (learning rate 10^{-4}, weight decay $5 \cdot 10^{-5}$, training batch size 256). In addition, 20% of the data was used for validation.

Algorithm 1: Sourcerer

Inputs: Source model q_ϕ constrained on the bounded domain B_Θ, observed dataset $\mathcal{D} = \{x_1, ..., x_n\} \sim p_o(x)$, differentiable model $p(x|\theta)$ to draw samples from (simulator or surrogate), number of samples m to estimate entropy, regularization schedule $\lambda_{t=1}, ..., \lambda_{t=T}$.
Outputs: Trained source model $q_\phi(\theta)$.

$t \leftarrow 0$;
while *not converged* **do**
  $\theta_1, \ldots, \theta_n \sim q_\phi(\theta)$; `# sample parameters for pushforward`
  $x'_i \sim p(x|\theta_i)$; `# sample pushforward`
  $\theta'_1, \ldots, \theta'_m \sim q_\phi(\theta)$; `# sample parameters for entropy estimation`
  $\lambda \leftarrow \lambda_{t=t}$ **if** $t \leq T$ **else** $\lambda_{t=T}$; `# schedule lambda`
  $\mathcal{L} \leftarrow \lambda H(\{\theta'_1, \ldots, \theta'_m\}) + (1-\lambda) D(\{x_1, \ldots x_n\}, \{x'_1, \ldots, x'_n\})$; `# compute loss`
  $\phi \leftarrow \phi - \text{Adam}(\nabla_\phi \mathcal{L})$; `# update source model`
  $t \leftarrow t + 1$
return q_ϕ

To train surrogate models for the SIR and Lotka-Volterra model, we first reduce the simulator dimension in observation space to 25 in both cases. Additionally, we add a small amount of independent Gaussian noise ($\mathcal{N}(X, 0.01^2)$) to the output of the SIR simulator to avoid training the normalizing flow surrogate with simulations from a deterministic likelihood. We then use 10^6 simulations to train and validate (20% validation set) both surrogate models, again using the Adam optimizer (learning rate $5 \cdot 10^{-4}$, weight decay $5 \cdot 10^{-5}$, training batch size 256).

A.6 Kozachenko-Leonenko entropy estimator

Our use of neural samplers requires us to use a sample-based estimate of (differential) entropy, since no tractable likelihood is available (see Sec. 2.5).

We use the Kozachenko-Leonenko estimator [28, 3] for a set of samples $\{\theta_i\}_{i=1}^n$ from a distribution $p(\theta) \in P(\Theta)$, given by

$$H(q_\phi) \approx \frac{d}{m} \left[\sum_{i=1}^{n} \log(d_i) \right] - g(k) + g(n) + \log(V_d), \tag{8}$$

where d_i is the distance of θ_i from its k-th nearest neighbor in $\{\theta_j\}_{j \neq i}$, d is the dimensionality of Θ, m is the number of non-zero values of d_i, g is the digamma function, and V_d is the volume of the unit ball using the same distance measure as used to compute the distances d_i.

The Kozachenko-Leonenko estimator is differentiable and can be used for gradient-based optimization. The all-pairs nearest neighbor problem can be efficiently solved in $\mathcal{O}(n \log n)$ [57]. In practice, we find all nearest neighbors by computing all pairwise distances on a fixed number of samples. Throughout all experiments, 512 source distribution samples were used to estimate the entropy during training.

A.7 Uniqueness of maximum entropy source distribution

Here, we prove the uniqueness of the maximum entropy source distribution (Proposition 2.1). First, however, we demonstrate for a simple example that the source distribution without the maximum entropy condition is not unique.

Example of non-uniqueness Consider the (deterministic) simulator $x = f(\theta) = |\theta|$. Further assume that our observed distribution is the uniform distribution $p(x) = \mathcal{U}(x; a, b)$, where $0 < a < b$. Due the symmetry of f, the source distribution $p(\theta)$ for the observed distribution $p(x)$ is not unique. Any convex combination of form $\alpha u_1(\theta) + (1-\alpha) u_2$, where $u_1(\theta) = \mathcal{U}(\theta; -b, -a)$ and $u_2(\theta) = \mathcal{U}(\theta; a, b)$ and $\alpha \in [0, 1]$ provides a source distribution. The maximum entropy source distribution is unique and is attained if both distributions are weighted equally with $\alpha = 0.5$.

Proof of Proposition 2.1 First, let us state Proposition 2.1 in full:

Let $\Theta \subset \mathbb{R}^{d_\Theta}$ and $\mathcal{X} \subset \mathbb{R}^{d_\mathcal{X}}$ be the parameter and observation spaces, respectively. Suppose that Θ is compact. Let $\mathcal{P}(\Theta) \subset L^1(\Theta)$ and $\mathcal{P}(\mathcal{X}) \subset L^1(\mathcal{X})$ be the set of probability measures on Θ and $\mathcal{X}$ respectively. Let $Q = \{q | q^\# = p_o \text{ almost everywhere } \} \subset \mathcal{P}(\Theta)$ be the set of source distributions for a given likelihood $p(x|\theta)$ and data distribution $p_o \in \mathcal{P}(\mathcal{X})$. Suppose that Q is non-empty and compact (in the L^1 norm topology). Then $q^* = \arg\max_{q \in Q} H(q)$ exists and is unique.

First, by the compactness assumption on Θ, the (differential) entropy of all $q \in P(\Theta)$ is bounded above (by the entropy of the uniform distribution on Θ), and so in particular it is finite. By the compactness assumption on Q, the entropy achieves its supremum of Q, that is, there exists a q^* such that $H(q^*) = \arg\max_{q \in Q} H(q)$. To show that q^* is unique (up to L^1-null sets), it is sufficient to show two results: (1) that the set Q is a convex set, and (2) that entropy is strictly concave. In this case, if we have two distinct suprema q_1^* and q_2^*, then any convex combination of q_1^*, q_2^* is a valid source distribution with higher entropy, causing a contradiction. For the remainder of this proof, we let q_1 and q_2 be two distinct source distributions. Their convex combination $q = \alpha q_1 + (1-\alpha) q_2$, $\alpha \in [0, 1]$ is a valid probability distribution supported on both of the supports of q_1 and q_2.

(1) *Sources distributions are closed under convex combination*: q is also a source distribution, since

$$\begin{aligned} q^\#(x) &= \int p(x|\theta) \cdot (\alpha q_1(\theta) + (1-\alpha) q_2(\theta)) d\theta \\ &= \alpha \int p(x|\theta) q_1(\theta) d\theta + (1-\alpha) \int p(x|\theta) q_2(\theta) d\theta \\ &= \alpha p_o(x) + (1-\alpha) p_o(x) = p_o(x). \end{aligned} \tag{9}$$

(2) *Entropy is (strictly) concave*: the entropy of q satisfies

$$\begin{aligned} H(q) &= -\int (\alpha q_1(\theta) + (1-\alpha) q_2(\theta)) \cdot \log(\alpha q_1(\theta) + (1-\alpha) q_2(\theta)) d\theta \\ &\geq -\int [\alpha q_1(\theta) \log(q_1(\theta)) + (1-\alpha) q_2(\theta) \log(q_2(\theta))] d\theta \\ &= \alpha H(q_1) + (1-\alpha) H(q_2), \end{aligned} \tag{10}$$

where we used the fact that the function $f(x) = x \log x$ is convex on $[0, \infty)$, and hence $-f$ is concave. Furthermore, $f(x)$ is strictly convex on $[0, \infty)$, so for any $\theta \in \Theta$, the equality of the integrands

$$\alpha q_1(\theta) + (1-\alpha) q_2(\theta)) \log(\alpha q_1(\theta) + (1-\alpha) q_2(\theta)) = \alpha q_1(\theta) \log(q_1(\theta) + (1-\alpha) q_2(\theta) \log(q_2(\theta) \tag{11}$$

holds if and only if $\alpha \in \{0, 1\}$ or $q_1(\theta) = q_2(\theta)$. Since q_1 and q_2 are assumed distinct, that is, it holds $q_1(\theta) \neq q_2(\theta)$ on a positive measure set, the integral equality in Eq. (10) only holds if $\alpha \in \{0, 1\}$, and thus entropy is strictly concave, which concludes our proof.

□

Regularized regression as an approximation to constrained optimization In practice, we approximate the optimization problem in Eq. (2) with the regularized regression objective in Eq. (3). As a result, we cannot use the result of Proposition 2.1 to guarantee the uniqueness of our solution. However, the dynamic schedule approach to λ we use in our work (see Appendix A.3) is similar to the penalty method of approximating solutions to constrained optimization tasks [16, 8]. Future work could use this connection to apply theoretical knowledge of constrained optimization in the source distribution estimation setting.

A.8 Examples related to the average posterior distribution

In general, the average posterior distribution is not a source distribution. The average posterior distribution is defined in Eq. (7). The infinite data limit is given by $G_n(\theta) \xrightarrow{n \to \infty} G(\theta) = \int p(\theta|x) p_o(x) dx$.

Here, we provide two examples, one based on coin flips, and one based on a Gaussian bimodal likelihood to illustrate this point.

Coin-flip example Consider the classical coin flip example, where the probability of heads (H) follows a Bernoulli distribution with parameter θ. The source distribution estimation problem for this setting would consist of the outcomes of flipping n distinct coins, with potentially different values θ_i.

Proposition A.1. *Suppose we have a Beta prior distribution on the Bernoulli parameter* $\theta \sim Beta(\alpha, \beta)$ *with parameters* $\alpha = \beta = 1$*, and that the empirical measurements consist of* 70% *heads, i.e.:*

$$p_o(x) = \begin{cases} 0.7 & x = H \\ 0.3 & x = T \end{cases}$$

Then the average posterior $G(\theta) = \int p(\theta|x)p_o(x)dx$ *is* not *a source distribution for* $p_o(x)$.

Proof: Since the Beta distribution is the conjugate prior for the Bernoulli likelihood, the single-observation posteriors are known to be $p(\theta|x = \text{H}) = Beta(2, 1)$ and $p(\theta|x = \text{T}) = Beta(1, 2)$. Hence, the average posterior is

$$G(\theta) = 0.3 \cdot Beta(1, 2) + 0.7 \cdot Beta(2, 1). \tag{12}$$

However, the ratio of heads observed when pushing this distribution through the Bernoulli simulator is

$$\begin{aligned} G^{\#}(x = \text{H}) &= \int_0^1 \theta[0.3 \cdot Beta(\theta; 1, 2) + 0.7 \cdot Beta(\theta; 2, 1)]d\theta \\ &= \int_0^1 \theta \left[0.3\frac{1-\theta}{B(1,2)} + 0.7\frac{\theta}{B(2,1)}\right] d\theta \\ &= 2\int_0^1 [0.3\theta(1-\theta) + 0.7\theta^2]d\theta \\ &= 0.3\theta^2 + \frac{2}{3}0.4\theta^3\Big|_0^1 \approx 0.567 \neq 0.7, \end{aligned} \tag{13}$$

where we have used the fact that the Beta function takes the values $B(1, 2) = B(2, 1) = 1/2$. Therefore, the pushforward of the average posterior distribution does not recover the correct ratio of heads, and so it is not a source distribution.

Gaussian bimodal example As another illustrative example to show the differences between average posterior and estimated source, we consider a one-dimensional, bimodal Gaussian likelihood given by $x|\theta \sim 0.5\mathcal{N}(x|\theta - 1, 0.3^2) + 0.5\mathcal{N}(x|\theta + 1, 0.3^2)$ and the source $\mathcal{N}(\theta|0, 0.25^2)$. We use the `sbi` package [54] and perform neural posterior estimation with the uniform prior $\theta \sim \mathcal{U}([-5, 5])$ to obtain the average posterior and compare it to the source estimated with our approach.

While the estimated source matches the original source closely, the average posterior is visibly different and substantially broader (Fig. A1). As expected, this difference persists when sampling from the average posterior and estimated source to simulate from the likelihood. The pushforward distributions in data space of the original and estimated source match, while the one of the average posterior is again substantially different (Fig. A1).

Additional average posteriors (in comparison to original and estimated source distributions) for the Two Moons and Gaussian mixture are shown in Fig. A6.

A.9 Details on source estimation for the single-compartment Hodgkin-Huxley model

We use the simulators as described in Bernaerts et al. [2] for our source estimation. This work provides a uniform prior over a specified box domain, which we use as the reference distribution for source estimation. Since the simulator parameters live on different orders of magnitude, we transform the original m-dimensional box domain to the $[-1, 1]^m$ cube. Note that this transformation does not affect the maximum entropy source distribution. This is because this scaling results in a constant term added to the (differential) entropy. More specifically, for a random variable X (associated with

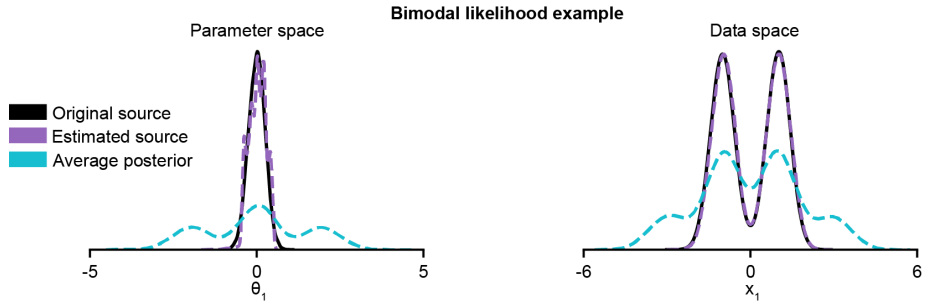


Figure A1: Failure of the average posterior as a source distribution for the bimodal likelihood example. Each of the individual posteriors is bimodal, resulting in an average posterior with 3 modes (left), the secondary modes produce observations which are not observed in the data distribution when pushed through the likelihood (right), and should not be part of the source distribution.

its probability density $p(x)$), the (differential) entropy of X scaled by a (diagonal) scaling matrix D and shifted by a vector c is given by

$$H(DX + c) = H(X) + \log(\det D). \tag{14}$$

The surrogate is trained on 10^6 parameter-simulation pairs produced by sampling parameters from the uniform distribution and simulating with the sampled parameters. We do not use the simulated traces directly, but instead compute 5 commonly used summary statistics [2, 18]. These are the number of spikes k transformed by a $\log(k + 3)$ transformation (ensuring it is defined in the case of $k = 0$), the mean of the resting potential, and the first three moments (mean, variance, and skewness) of the voltage during the stimulation.

As our surrogate, we choose a deterministic multi-layer perceptron, because we found that the internal noise has almost no noticeable effect on the summary statistics, so that the likelihood $p(x|\theta)$ is essentially a point function. We are able to make this choice because the sample based nature of our source distribution estimation approach is less sensitive to sharp likelihood functions, whereas likelihood-based approaches could struggle with such problems.

The multi-layer perceptron (MLP) surrogate has 3 layers with a hidden dimension of 256. ReLU activations and batch normalization were used. Training of the MLP was done with Adam (learning rate $5 \cdot 10^{-4}$, weight decay 10^{-5}, training batch size 4096). Again, 20% of the data were used for validation.

A.10 Computational Resources

All numerical experiments reported in this work were performed on GPU using an NVIDIA A100 GPU. A single source estimation run for a benchmark task using the Sourcerer approach (for one value of λ) took approx. 30 seconds. In comparison, learning the source using NEB for the same task took approx. 2 minutes (see Table A1). A source estimation run for Sourcerer on the high-dimensional tasks took approx. 10 min. When the observations are high-dimensional, training a surrogate (if required) makes up the majority of the computational cost. For the Hodgkin-Huxley task, training a surrogate took approx. 20 minutes, after which estimating the source distribution with Sourcerer took approx. 30 seconds.

Table A1: **Wall-clock runtime comparison between Sourcerer and NEB.** Time in seconds measured on an Nvidia A100 GPU. Average and standard deviation are shown over 5 runs. For all three settings (Sourcerer with and without entropy regularization, NEB), surrogate models for the benchmark simulators were used. Sourcerer converges noticeably faster than the NEB baseline.

Method	Sur. (w/o reg.)	Sur. (with reg.)	NEB
TM	29.4 (8.5)	63.9 (10.1)	145.2 (13.9)
IK	28.5 (6.9)	66.7 (10.0)	116.8 (22.6)
SLCP	71.7 (12.8)	53.1 (12.2)	91.6 (9.9)
GM	26.6 (5.4)	46.2 (9.2)	98.5 (15.5)

A.11 Supplementary figures

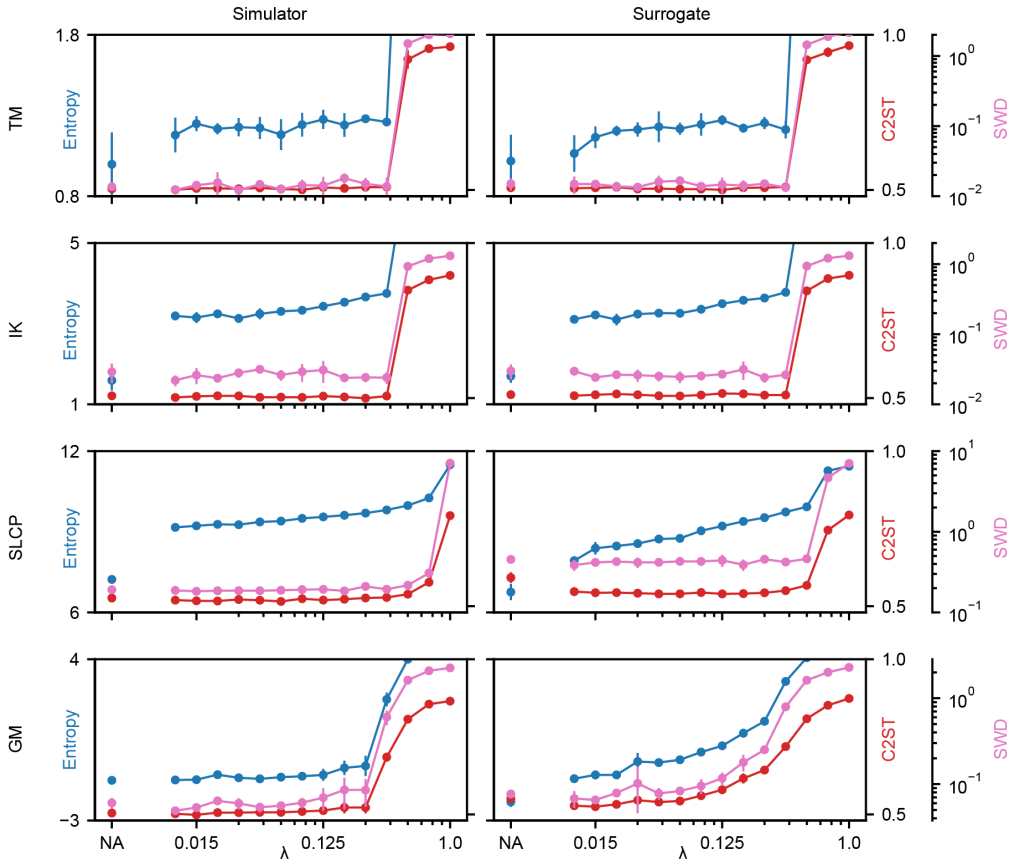


Figure A2: Extended results for source distribution estimation on the benchmark tasks (Fig. 3) for different choices of λ. In addition to the C2ST accuracy and entropy, here the Sliced-Wasserstein distance (SWD) between the observations and the pushforward distribution of the estimated source is shown. Mean and standard deviation were computed over five runs.

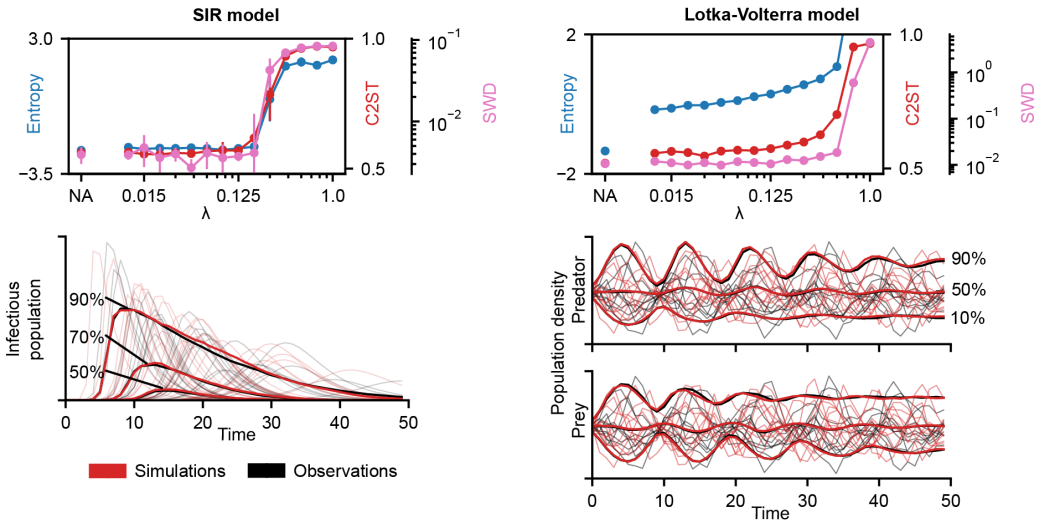


Figure A3: Extended results for source distribution estimation on the differentiable SIR and Lotka-Volterra models (Fig. 4). In addition to the Sliced-Wasserstein distance (SWD), the C2ST accuracy between the observations and the pushforward distribution of the the estimated source is shown. Despite the high-dimensional data space of the simulators (50 and 100 dimensions), the estimated sources achieve a good C2ST accuracy (below 60%) for various choices of λ. Mean and standard deviation were computed over five runs. Additionally, percentile values of all samples computed per time point between simulations (simulated with parameters from the estimated source) and observations (simulated with parameters from the original source) closely match.

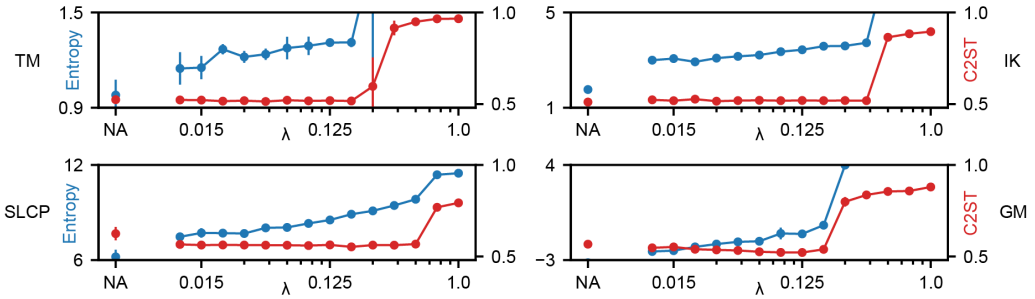


Figure A4: Sourcerer with Maximum Mean Discrepancy (MMD) as the differentiable, sample-based distance. We use MMD with an RBF kernel and the median distance heuristic for selecting the kernel length scale. Source estimation is performed without (NA) and with entropy regularization for different choices of λ. For these tasks, MMD produces similar results to the previously used SWD (Fig. 3b). These results show that Sourcerer is compatible with other sample-based, differentiable distances other than the SWD. For all cases, mean C2ST accuracy between observations and simulations (lower is better) as well as the mean entropy of estimated sources (higher is better) over five runs are shown together with the standard deviation.

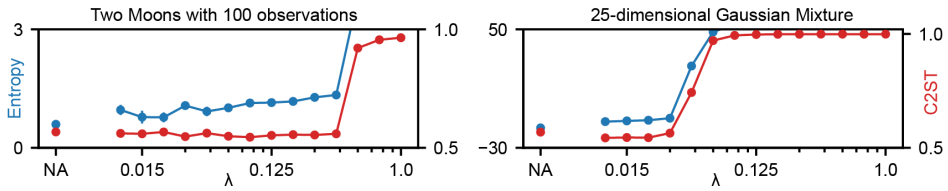


Figure A5: Experiments with less observations and higher-dimensional sources. Source estimation without (NA) and with entropy regularization for different choices of λ. For the Two Moons task, the number of observations was reduced from 10000 to 100. For the Gaussian Mixture task, the dimensionality was increased from 2 to 25. These results show that Sourcerer is robust to small datasets of observations, and can estimate high-dimensional source distributions. For all cases, mean C2ST accuracy between observations and simulations (lower is better) as well as the mean entropy of estimated sources (higher is better) over five runs are shown together with the standard deviation.

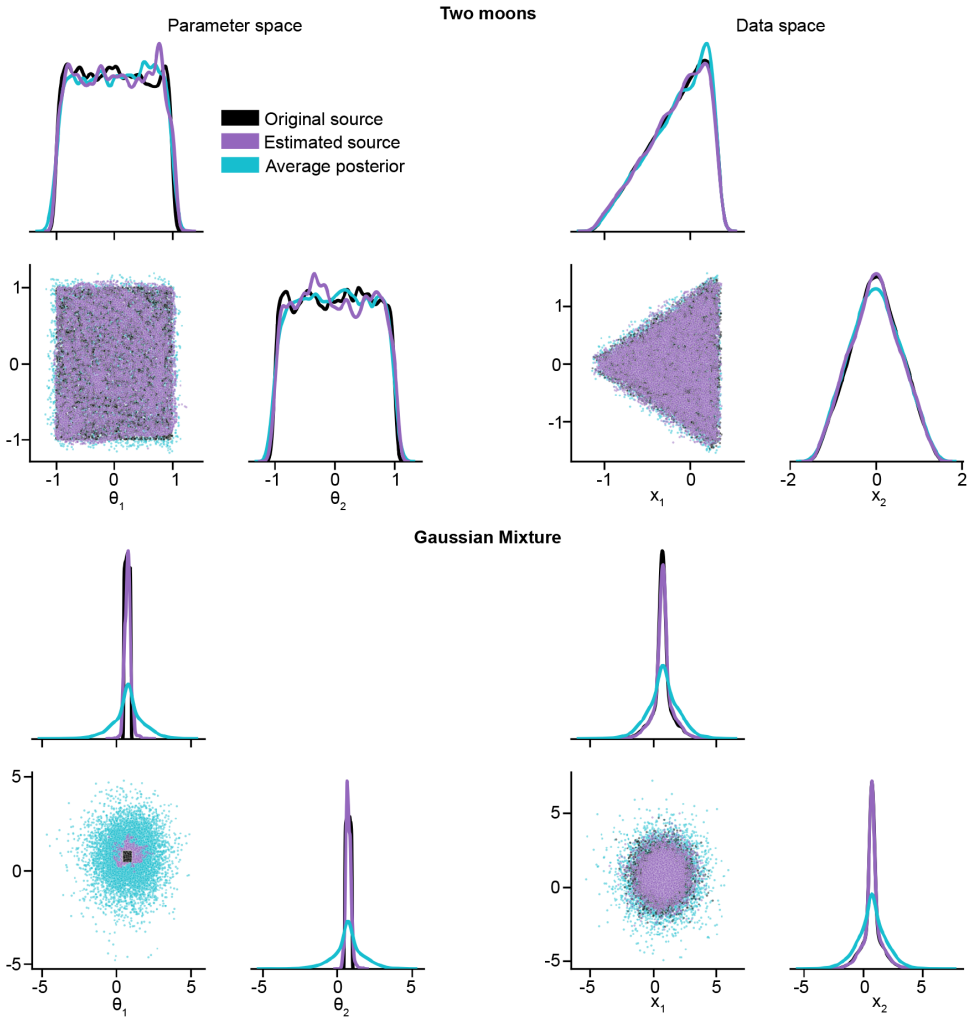


Figure A6: Original and estimated sources distributions as well as average posterior distribution for Two Moons and Gaussian Mixture simulator with uniform prior $\theta \sim \mathcal{U}([-5, 5]^2)$. For simulators for which the likelihood is unimodal and narrow, such as the Two Moons simulator, the average posterior can be a good approximation of a source distribution. However, for simulators where the likelihood is broader, such as the Gaussian Mixture simulator, the average posterior is too broad, and does not reproduce the data distribution p_o well, when compared to estimates of source distributions.

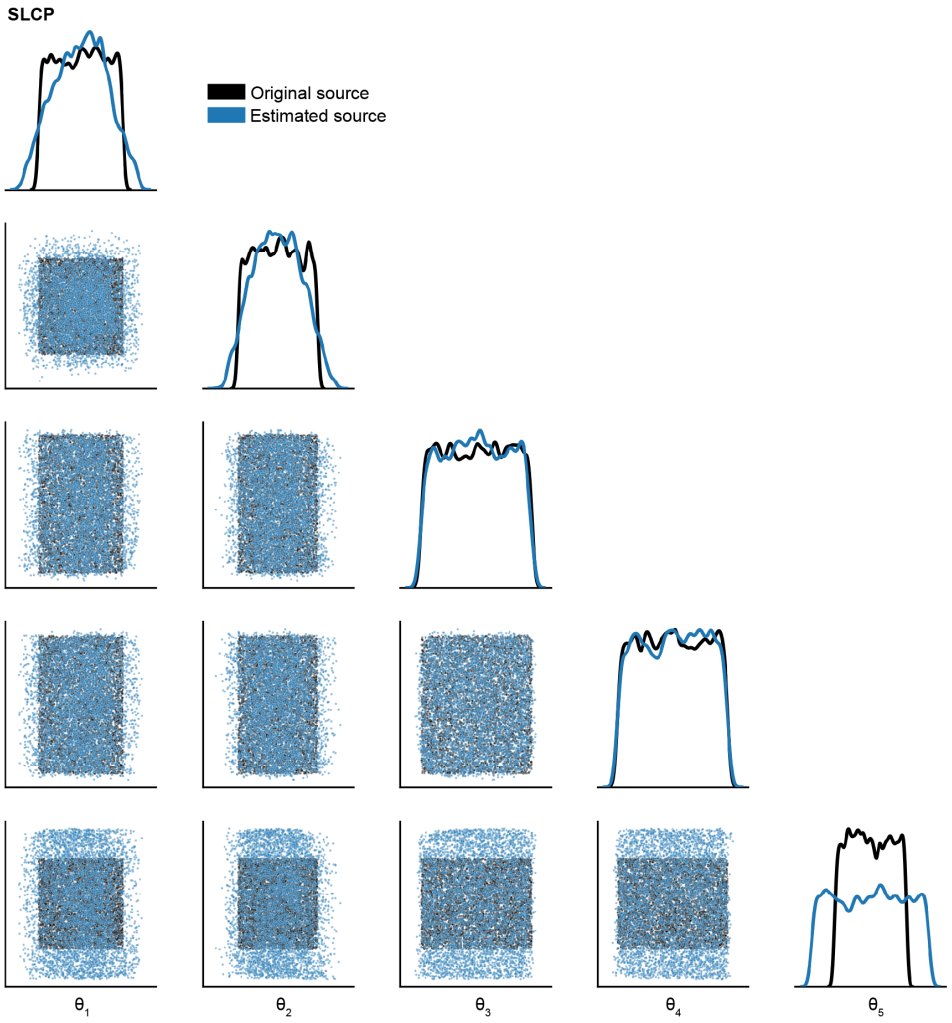


Figure A7: Original and estimated source distributions for the benchmark SLCP simulator. The estimated source has higher entropy than the original source.

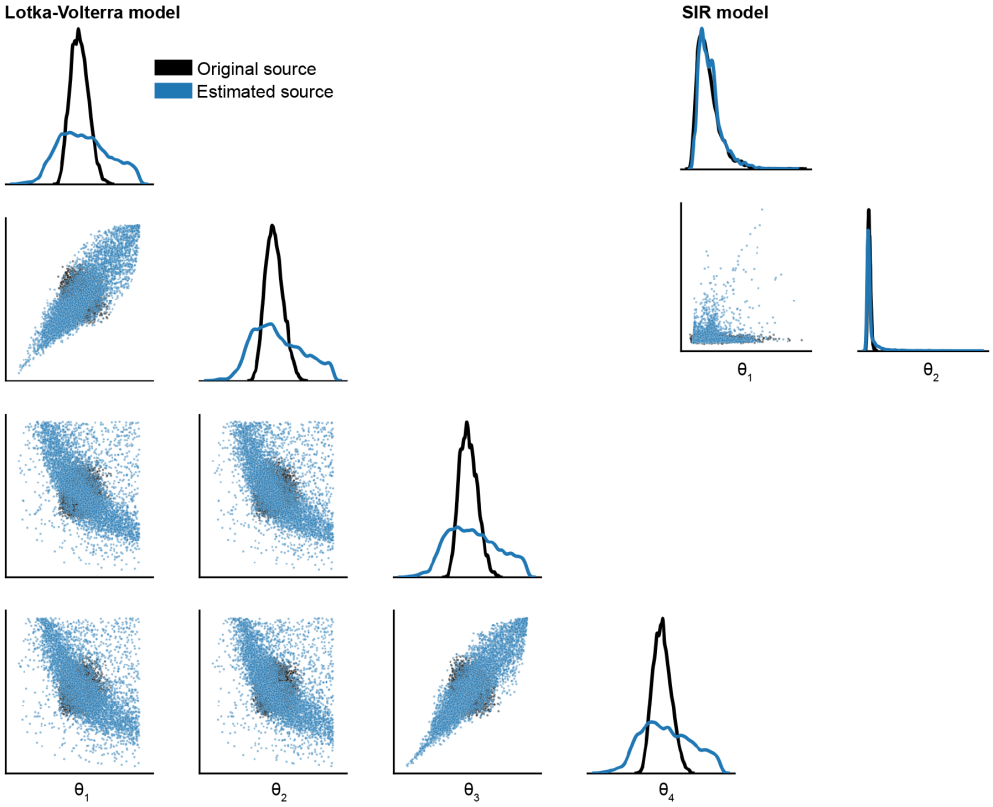


Figure A8: Original and estimated source distributions for the SIR and Lotka-Volterra model. For the Lotka-Volterra model, the estimated source has higher entropy than the original source.

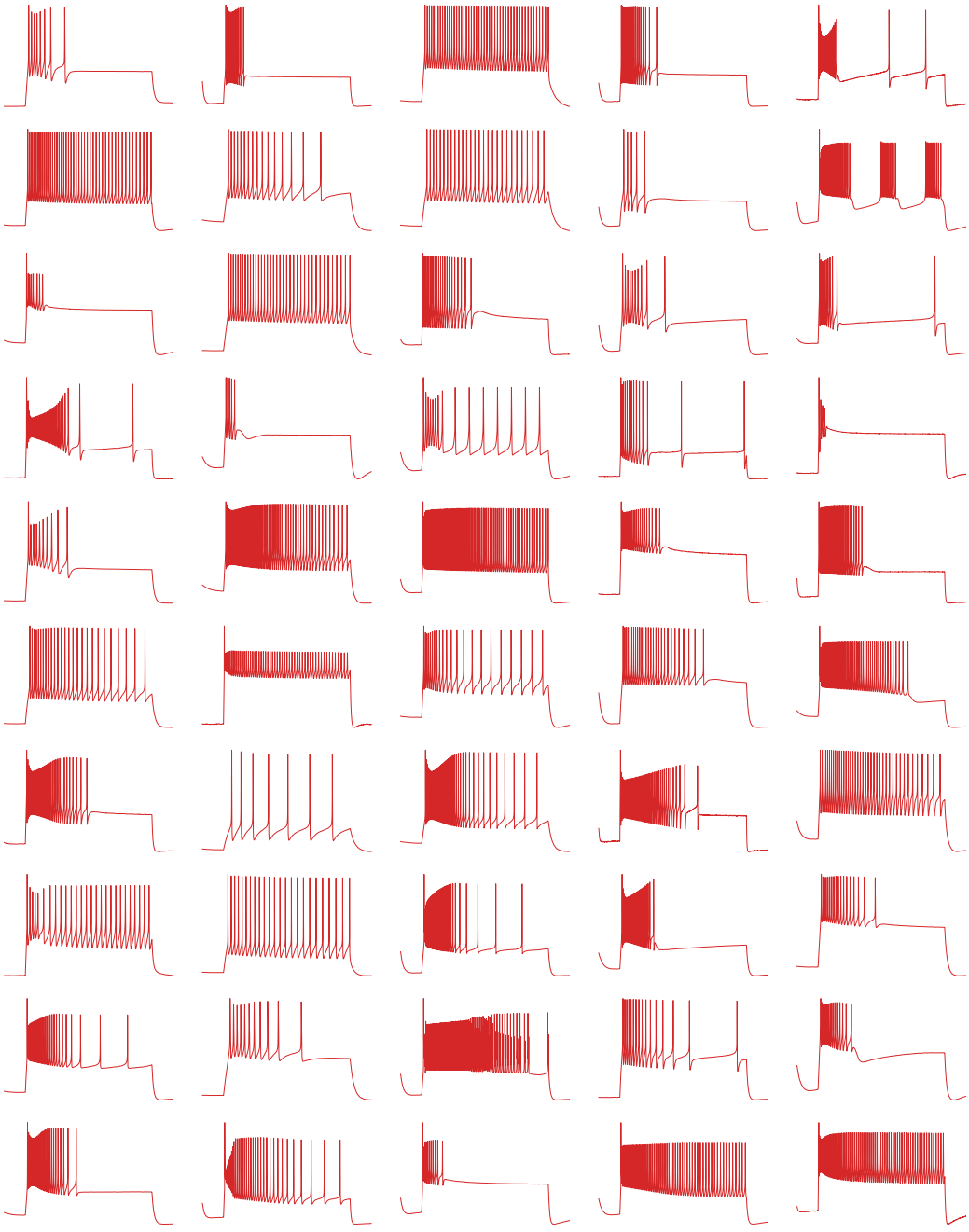


Figure A9: 50 random example traces produced by sampling from the estimated source and simulating with the Hodgkin-Huxley model.

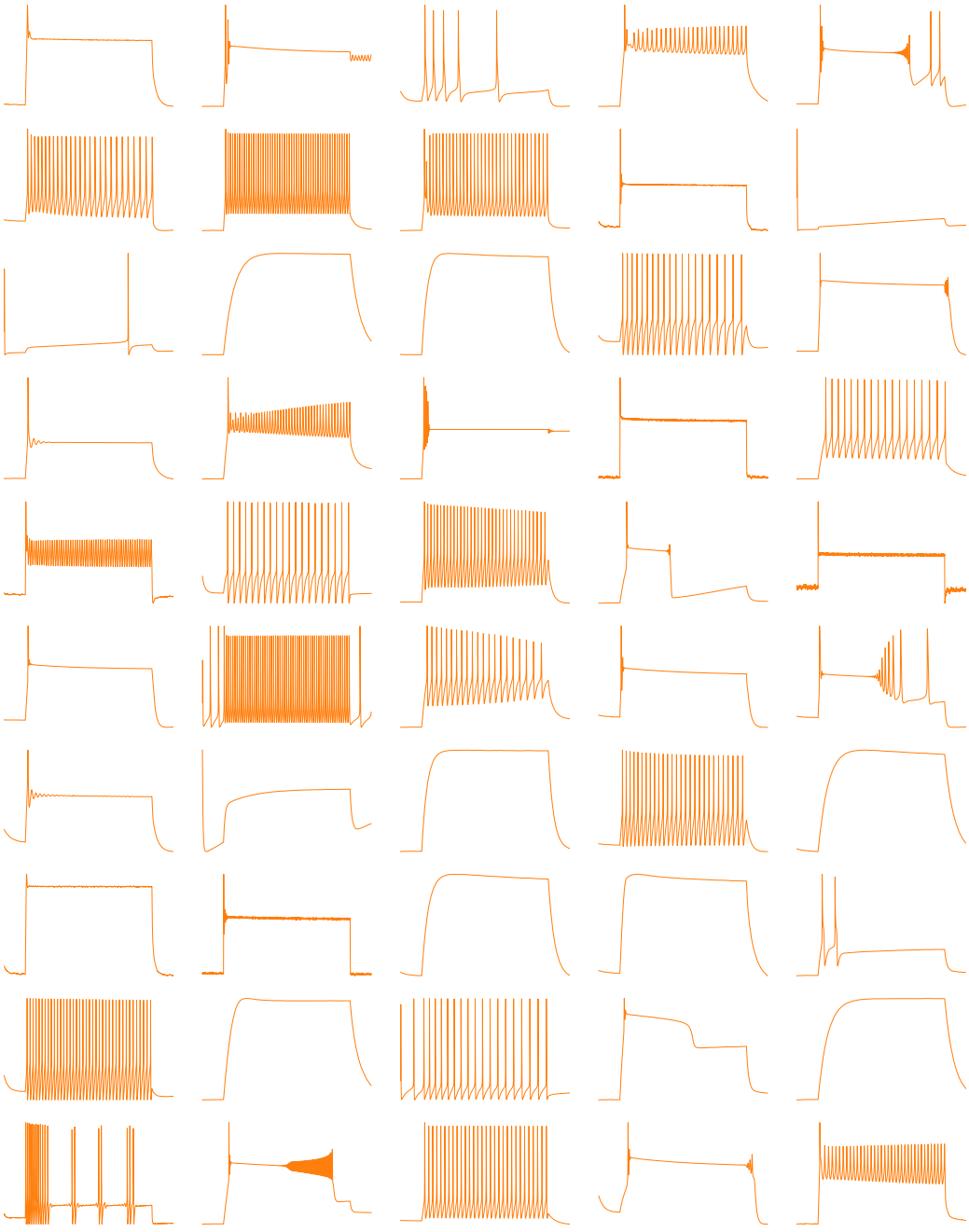


Figure A10: 50 random example traces produced by sampling from the uniform distribution over the box domain and simulating with the Hodgkin-Huxley model.

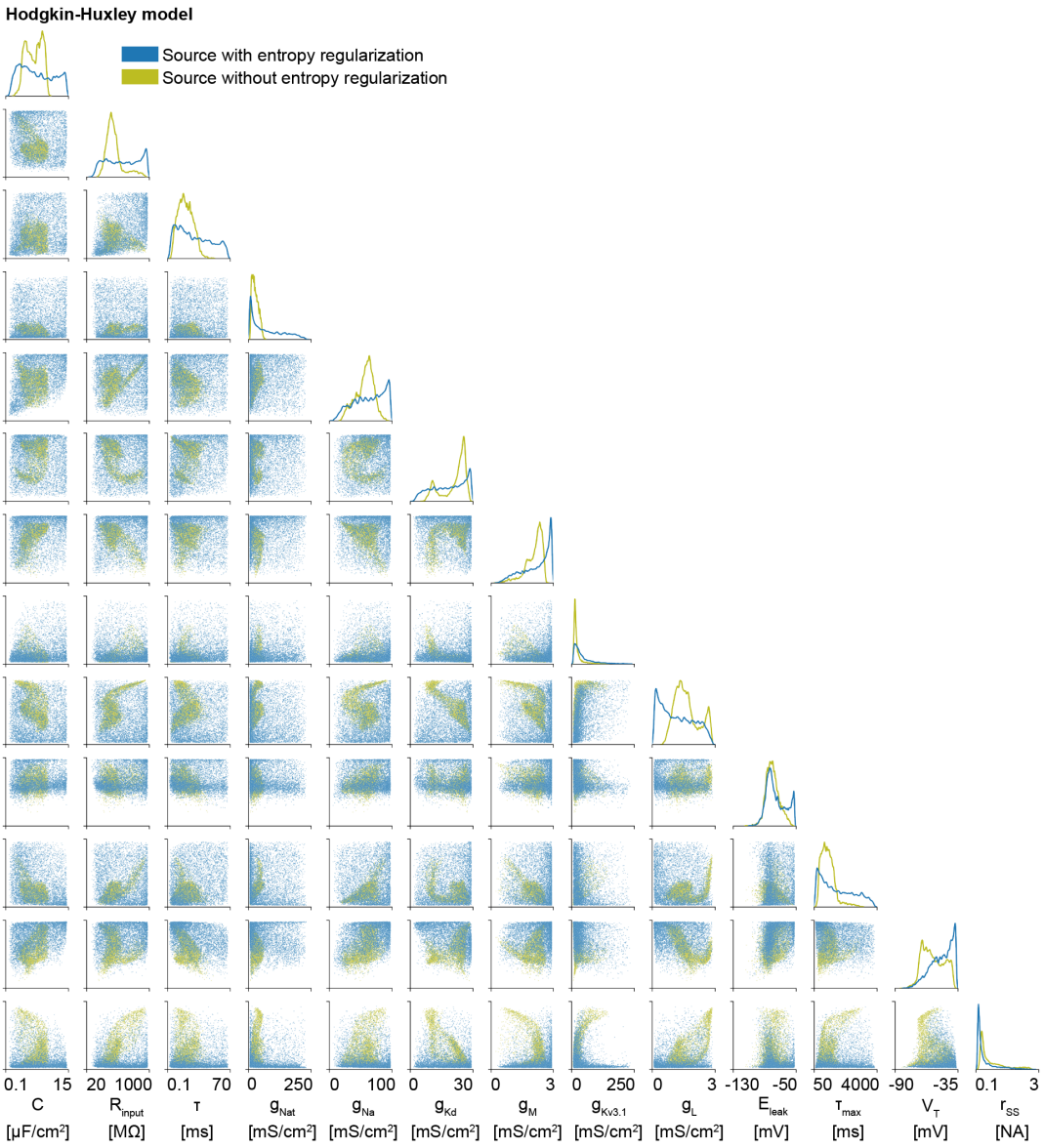


Figure A11: Estimated sources using for Hodgkin-Huxley task with the entropy regularization ($\lambda = 0.25$) and without the entropy regularization. Without, many viable parameter settings are missed, which would have significant downstream effects if the learned source distribution is used as a prior distribution for inference tasks.

Effortless, Simulation-Efficient Bayesian Inference using Tabular Foundation Models

Julius Vetter[1,2] * **Manuel Gloeckler**[1,2] * **Daniel Gedon**[1,2] † **Jakob H. Macke**[1,2,3] †

[1]Machine Learning in Science, University of Tübingen, Tübingen, Germany
[2]Tübingen AI Center, Tübingen, Germany
[3]Department Empirical Inference, Max Planck Institute for Intelligent Systems, Tübingen, Germany

Abstract

Simulation-based inference (SBI) offers a flexible and general approach to performing Bayesian inference: In SBI, a neural network is trained on synthetic data simulated from a model and used to rapidly infer posterior distributions for observed data. A key goal for SBI is to achieve accurate inference with as few simulations as possible, especially for expensive simulators. In this work, we address this challenge by repurposing recent probabilistic foundation models for tabular data: We show how tabular foundation models—specifically TabPFN—can be used as pre-trained autoregressive conditional density estimators for SBI. We propose Neural Posterior Estimation with Prior-data Fitted Networks (NPE-PFN) and show that it is competitive with current SBI approaches in terms of accuracy for both benchmark tasks and two complex scientific inverse problems. Crucially, it often substantially outperforms them in terms of simulation efficiency, sometimes requiring orders of magnitude fewer simulations. NPE-PFN eliminates the need for selecting and training an inference network and tuning its hyperparameters. We also show that it exhibits superior robustness to model misspecification and can be scaled to simulation budgets that exceed the context size limit of TabPFN. NPE-PFN provides a new direction for SBI, where training-free, general-purpose inference models offer efficient, easy-to-use, and flexible solutions for a wide range of stochastic inverse problems.

1 Introduction

Simulation has long been a cornerstone of scientific inquiry [1, 2] and is becoming increasingly relevant as researchers tackle ever more complex scientific questions and systems [3]. Simulators often depend on parameters that are challenging or impossible to measure experimentally. Bayesian inference provides a general framework for identifying such parameters by estimating posterior distributions over parameters. However, classical methods such as Markov chain Monte Carlo (MCMC) require evaluations of the associated model likelihoods, which can be computationally demanding or prohibitive for complex numerical simulators.

The field of simulation-based inference (SBI), or likelihood-free inference, aims to address this challenge by enabling Bayesian inference without requiring access to likelihoods. The basic idea of SBI is to train a neural network on synthetic data simulated from the model such that it can

*Equal contribution.
†Joint supervision.
{firstname.lastname}@uni-tuebingen.de
Code available at https://github.com/mackelab/npe-pfn.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

approximate the posterior distributions. In Neural Posterior Estimation (NPE, [4, 5]), a conditional density neural network—often, a normalizing flow [6, 7] or a diffusion model [8–10]—learns the conditional distribution of parameters given simulated data $p(\boldsymbol{\theta} \mid \boldsymbol{x})$. When evaluated on empirical observations $\boldsymbol{x}_o$, the network directly returns the posterior distribution $p(\boldsymbol{\theta} \mid \boldsymbol{x}_o)$. Similarly, other SBI methods use neural networks to represent likelihoods [11–13], likelihood ratios [14–17], or target several properties at once [18–21]. SBI has been used for scientific discovery in various domains, including astrophysics [22] and neuroscience [23].

Compared to classical Approximate Bayesian Computation (ABC, [24, 25]) or synthetic likelihoods methods [26], neural-network-based SBI methods scale better to complex, high-dimensional simulators [27]. In addition, SBI methods often *amortize* the computational burden: Once trained, the network provides fast inference across multiple observations.

Despite these advances, a key challenge for SBI methods remains: They typically require generating a large number of simulations as synthetic training data, rendering them impractical for expensive simulators. Recent work aims to increase simulation efficiency, e.g., by exploiting additional information or properties of the simulator [7, 28–31], utilizing low-fidelity simulations [32, 33], or using Bayesian neural networks [34–36]. So-called *sequential* approaches forego the amortization properties of the inference network and target simulations to particular observations [4–6, 37], but even then, simulation efficiency remains a challenge. In addition, users must select an appropriate SBI method and select and train an inference network with the associated hyperparameters—posing a barrier for inexperienced users.

These limitations raise a key question: Can we eliminate the high cost associated with training and tuning, and alleviate the need to have a large number of simulations by leveraging recent advances in foundation models? Here, we answer this question by repurposing a tabular foundation model—specifically Tabular Prior-data Fitted Networks v2 (TabPFN) [38]—as an inference engine for SBI. TabPFN is trained to perform in-context learning on tabular data (i.e., structured data organized in rows and columns), returning a prediction when being prompted with a training dataset and a test point. For both regression and classification, TabPFN was shown to perform exceptionally well, in particular in low data regimes. Crucially, TabPFN can also serve as a density estimator by applying it in an autoregressive manner, though prior work has only explored this in limited settings [38].

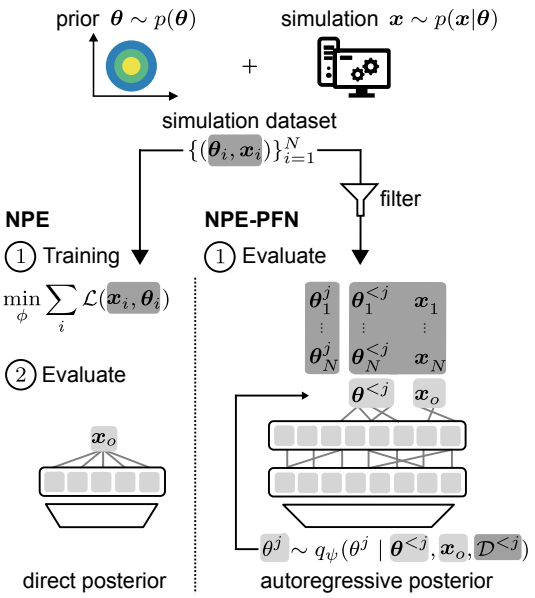


Figure 1: Comparison of NPE to NPE-PFN (ours): Both approaches use simulations sampled from the prior and simulator. In (standard) NPE, a neural density estimator is trained to obtain the posterior. In NPE-PFN, the posterior is evaluated by autoregressively passing the simulation dataset and observations to TabPFN.

Here, we show that using TabPFN autoregressively allows for the estimation of complex, high-dimensional conditional densities, making it suitable for SBI. Based on this observation, we introduce Neural Posterior Estimation with Prior-data Fitted Networks (NPE-PFN), a variant of NPE that uses TabPFN as a pre-trained conditional density estimator, and therefore enables inference without any additional neural network training (Fig. 1). On benchmark tasks, NPE-PFN performs competitively with other SBI approaches and often outperforms them substantially, especially for small simulation budgets. Furthermore, we leverage data-filtering schemes that enable NPE-PFN to make use of large simulation budgets exceeding the context size of TabPFN and extend the approach to sequential inference settings. Finally, we show that NPE-PFN remains empirically robust under model misspecification and achieves strong results on two challenging scientific applications, requiring orders of magnitude fewer simulations than standard SBI approaches.

2 Neural Posterior Estimation with Prior-data Fitted Networks

2.1 Background

We consider a simulator with parameters $\boldsymbol{\theta} \in \mathbb{R}^{d_\theta}$ that stochastically generates samples $\boldsymbol{x} \in \mathbb{R}^{d_x}$, thus implicitly defining a likelihood $p(\boldsymbol{x} \mid \boldsymbol{\theta})$. Given a prior distribution $p(\boldsymbol{\theta})$ and observation $\boldsymbol{x}_o$, our aim is to infer the associated posterior distribution $p(\boldsymbol{\theta} \mid \boldsymbol{x}_o) \propto p(\boldsymbol{x}_o \mid \boldsymbol{\theta})p(\boldsymbol{\theta})$.

Simulation-based inference. We focus on Neural Posterior Estimation (NPE), a common approach to SBI: NPE approximates the target posterior distribution directly by training an inference network $q_\phi(\boldsymbol{\theta} \mid \boldsymbol{x})$ on a dataset of parameter–data pairs $\mathcal{D} = \{(\boldsymbol{\theta}_i, \boldsymbol{x}_i)\}_{i=1}^N$ sampled from a prior $p(\boldsymbol{\theta})$ and a simulator with implicit likelihood $p(\boldsymbol{x} \mid \boldsymbol{\theta})$. The inference network is trained to maximize the log-likelihood and learns to approximate the corresponding posterior distribution [4, 39]. Flexible conditional neural density estimators, such as normalizing flows [40–42], are typically used as inference networks. At inference time, NPE provides an amortized posterior approximation, allowing for rapid evaluations.

Tabular foundation models. Recent foundation models for tabular data have shown strong performance on regression and classification tasks via in-context learning, in which predictions for new inputs are made by conditioning the model on a context dataset [43]. We focus on Prior-data Fitted Networks (PFNs) [44, 45], specifically TabPFN [38], a state-of-the-art transformer-based PFN foundation model [46] for probabilistic classification and regression on tabular data. It has been pre-trained on synthetic datasets $\{(y_i, \boldsymbol{x}_i)\}_{i=0}^M$ with up to $M = 2048$ samples from randomly generated structural causal models by maximizing the likelihood $q_\psi(y_0 \mid \boldsymbol{x}_0, \mathcal{T})$, where $\mathcal{T} = \{(y_i, \boldsymbol{x}_i)\}_{i=1}^M$ is the in-context dataset. After training, TabPFN has been tested to perform well up to a feature dimension of $d_x \leq 500$ and a context size of $N \leq 10^4$ [38].

At inference, TabPFN conditions on a given in-context dataset $\mathcal{T}'$ and test point $\boldsymbol{x}_o$ to estimate the distribution $p(y \mid \boldsymbol{x}_o) \approx q_\psi(y \mid \boldsymbol{x}_o, \mathcal{T}')$ over the target variable y. Importantly, in its basic version, TabPFN is limited to *univariate* targets ($y \in \mathbb{R}$ for regression, $y \in \{1, \dots, n\}$ for classification). Thus, TabPFN is an in-context density estimator for one-dimensional densities. We perform inference over a model parameter $\theta \in \mathbb{R}$ by setting $y = \theta$ to estimate the posterior distribution $p(\theta \mid \boldsymbol{x}_o) \approx q_\psi(\theta \mid \boldsymbol{x}_o, \mathcal{D})$ given an observation $\boldsymbol{x}_o$ and a dataset of simulations $\mathcal{D}$. However, to estimate full posterior distributions over multiple model parameters $\boldsymbol{\theta} \in \mathbb{R}^{d_\theta}$, conditional density estimation in high dimensions is required.

2.2 TabPFN for simulation-based inference

To perform simulation-based inference with TabPFN, we repurpose it as a general conditional density estimator. The key to estimating high-dimensional densities with TabPFN is to use it autoregressively, i.e., to use it to sequentially predict the next dimension of the data with the previously processed dimensions and covariates in context.

In SBI, given a prior $p(\boldsymbol{\theta})$ and simulator $p(\boldsymbol{x} \mid \boldsymbol{\theta})$, the multivariate posterior distribution $p(\boldsymbol{\theta} \mid \boldsymbol{x}_o)$ for observation $\boldsymbol{x}_o$ can be decomposed as

$$p(\boldsymbol{\theta} \mid \boldsymbol{x}_o) \approx \prod_{j=1}^{d_\theta} q_\psi(\theta^j \mid \boldsymbol{\theta}^{<j}, \boldsymbol{x}_o, \mathcal{D}^{<j})$$

where $\mathcal{D}^{<j} = \{\theta_i^j, [\boldsymbol{\theta}_i^{<j}, \boldsymbol{x}_i]\}$ with $<j$ denoting the vector from index 1 up to, but not including j. Thus, using an arbitrary but fixed order of the parameter dimensions, one evaluation of the posterior $p(\boldsymbol{\theta} \mid \boldsymbol{x}_o)$ requires d_θ evaluations of TabPFN, making the prediction more expensive as the dimensionality increases. However, unlike standard NPE methods, this approach bypasses model fitting entirely and directly evaluates the posterior. We refer to our proposed approach as NPE-PFN, in contrast to standard NPE, which requires training an inference network, e.g., a normalizing flow (Fig. 1, pseudocode for NPE-PFN in Appendix Alg. 1).

2.3 Increasing the effective context size

Standard NPE can approximate arbitrarily complex conditional densities, given enough simulations and a sufficiently expressive inference network [4]. In contrast, NPE-PFN is limited by TabPFN's

context size of approximately 10^4 samples, beyond which additional data yields diminishing performance gains. While we expect NPE-PFN to be particularly powerful for *small* simulation counts, it is nevertheless desirable to also be able to work with larger simulation budgets using in-context learning.

Filtering simulations based on relevance. To overcome the limited context size, previous work proposed to use a subset of the training data containing only the nearest neighbors of a given test point as the context dataset [47]. Here, we show that this procedure, and generalizations of it, are sound from a Bayesian perspective when estimating the full posterior distribution. To this end, we exploit a key property of (neural) posterior estimation: The posterior can be estimated using only samples that are very close to the observation $\boldsymbol{x}_o$. More specifically, for any non-negative function $f_{\boldsymbol{x}_o}$ that satisfies $f_{\boldsymbol{x}_o}(\boldsymbol{x}_o) > 0$, we can reweigh the joint distribution $p(\boldsymbol{x}, \boldsymbol{\theta}) f_{\boldsymbol{x}_o}(\boldsymbol{x})$ without affecting the posterior distribution at convergence (details in Appendix Sec. B.2). This property allows us to *filter* simulations based on relevance without biasing the estimation. Choosing the filter as $f_{\boldsymbol{x}_o}(\boldsymbol{x}) = \mathbb{I}(d(\boldsymbol{x}, \boldsymbol{x}_o) < \epsilon)$, where $d(\cdot, \cdot)$ is a suitable distance metric and $\epsilon > 0$ is a threshold, effectively recovers an ABC-like selection scheme, where only simulations close to $\boldsymbol{x}_o$ are retained. In practice, ϵ is set adaptively to include the N_{filter} closest simulations, ensuring that the full context is used. Here, we consider the Euclidean distance $d(\boldsymbol{x}_o, \boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{x}_o\|_2$ after standardizing each feature dimension. As noted above, this filtering based on nearest neighbors has previously been explored and is also referred to as *localization* or *retrieval* [47–49]. This filtering principle also underlies various SBI methods, such as the aforementioned ABC [24] or calibration kernels in NPE training [5].

Filtering for simulations that are relevant or close to a given observation makes the (context) dataset dependent on that observation. For classical SBI methods, this step would require re-training the density estimator for each new observation, thus breaking amortization. However, NPE-PFN remains amortized: Since it is training-free, inference for new observations requires only a computationally negligible filtering step over a fixed set of simulations. Finally, filtering only works for *conditional* density estimation problems such as (simulation-based) Bayesian inference, which is our focus here. However, we also show how (Appendix Sec. B.4) this approach can be extended to *unconditional* density estimation on a larger number of samples. In the unconditional case, we partition the data space into regions that can be solved with local contexts.

Embedding data. Similarly to context size, the feature size can be a limitation, particularly for high-dimensional observations. This limitation can be handled either by performing inference using human-crafted summary statistics (as commonly done in ABC) or using (pre-trained) embedding networks [5, 50]. In principle, NPE-PFN allows for training an embedding network end-to-end (e.g., [51] in the context of classification), but we here investigate pre-training an embedding network independently of the density estimator (Appendix Sec. B.3).

2.4 Truncated sequential NPE-PFN

To reduce the number of simulations, *sequential* SBI methods have been developed that acquire simulations adaptively in rounds for one observation $\boldsymbol{x}_o$. We focus on Truncated Sequential NPE (TSNPE) [37], which mitigates bias by exploiting the (approximate) invariance of the posterior under modifications outside of its high-density regions (HDR). Formally, let $\text{HDR}_p^{1-\alpha} := \{\boldsymbol{\theta} \mid p(\boldsymbol{\theta} \mid \boldsymbol{x}_o) > k_\alpha\}$, where k_α denotes the density threshold containing $\alpha\%$ of the probability mass. Then, given a truncated prior $\tilde{p}(\boldsymbol{\theta}) = \mathbb{I}(\boldsymbol{\theta} \in \text{HDR}_p^{1-\alpha}) p(\boldsymbol{\theta})$, the posterior will be approximately preserved for $\boldsymbol{x}_o$. In TSNPE, a density estimator is trained in rounds. Starting with the full prior in the first round, the trained posterior approximation $q(\boldsymbol{\theta} \mid \boldsymbol{x}_o)$ is used to estimate $\text{HDR}_p^{1-\alpha} \approx \text{HDR}_q^{1-\alpha}$ and to truncate the prior to the HDR of the posterior approximation for the next round. In each subsequent round, the newly truncated prior is used to generate new simulations adaptively and refine $q(\boldsymbol{\theta} \mid \boldsymbol{x}_o)$ through continued training. Notably, training becomes redundant by replacing the density estimator with the TabPFN regressor, to which we will refer as TSNPE-PFN. In this training-free setting, the main computational burden shifts to truncating the prior distribution. Prior truncation can be achieved either by approximate approaches, such as sampling-importance resampling, or exactly by rejection sampling from the prior, i.e., checking whether a prior sample falls within the HDR. However, the latter approach requires repeated density estimation with the TabPFN regressor to evaluate $q(\boldsymbol{\theta} \mid \boldsymbol{x}_o)$, which can be costly to do autoregressively, a problem we address next.

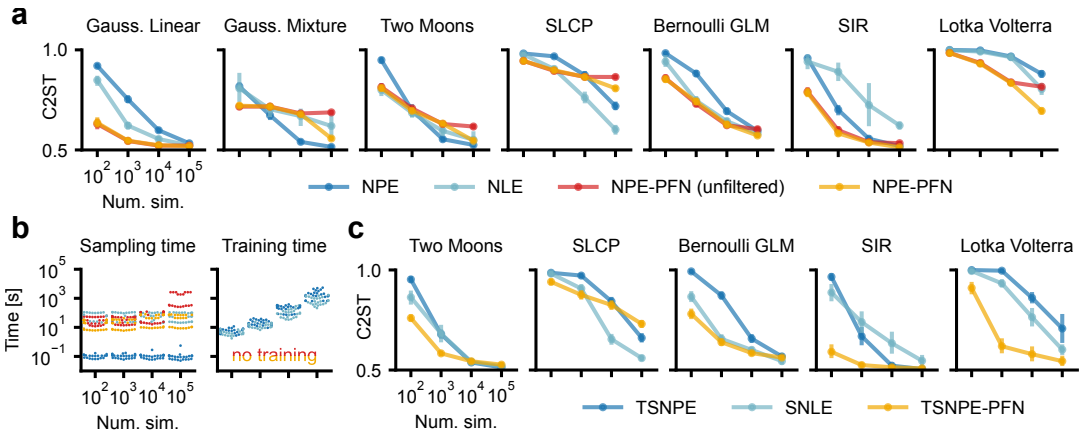


Figure 2: **SBI benchmark results for amortized and sequential NPE-PFN. (a)** C2ST for NPE, NLE, and NPE-PFN across ten reference posteriors (lower is better); dots indicate averages and bars show 95% confidence intervals over five independent runs. **(b)** Average time to generate 10^4 posterior samples and (if applicable) training time. **(c)** C2ST for sequential methods.

For rejection sampling, we need to evaluate $q(\boldsymbol{\theta} \mid \boldsymbol{x}_o)$ for a large number of prior samples $\boldsymbol{\theta} \sim p(\boldsymbol{\theta})$. In practice, the posterior is often much narrower than the prior and a large percentage of these samples will be rejected. This high rejection rate necessitates a fast, approximate approach for density evaluation, which we will construct using the "ratio-trick" [52]. We first sample an initial set of posterior samples $\{\boldsymbol{\theta}_i\}_i$ from the NPE-PFN posterior once and assign these samples the class label $y = 1$. We then contrast these samples with samples from a uniform distribution $\mathcal{U}(\boldsymbol{\theta}; \boldsymbol{\theta}_{min}, \boldsymbol{\theta}_{max})$ to which we assign the class label $y = 0$. The Bayes optimal classifier for this classification problem is given by

$$P_{\boldsymbol{x}_o}(y = 1 \mid \boldsymbol{\theta}) = \frac{p(\boldsymbol{\theta} \mid \boldsymbol{x}_o)}{p(\boldsymbol{\theta} \mid \boldsymbol{x}_o) + \mathcal{U}(\boldsymbol{\theta}; \boldsymbol{\theta}_{min}, \boldsymbol{\theta}_{max})}, \quad \text{thus,} \quad p(\boldsymbol{\theta} \mid \boldsymbol{x}_o) = \frac{P_{\boldsymbol{x}_o}(y = 1 \mid \boldsymbol{\theta})}{1 - P_{\boldsymbol{x}_o}(y = 1 \mid \boldsymbol{\theta})},$$

within the bounds $\boldsymbol{\theta}_{min}, \boldsymbol{\theta}_{max}$. After autoregressively sampling from the desired posterior to construct the training dataset, this approach, which we will refer to as *ratio-based* density evaluation, requires only a single forward pass using the TabPFN classifier and is, therefore, significantly faster than autoregressive density evaluation, especially as the parameter dimension d_θ or the number of required density evaluations grows (pseudocode for TSNPE-PFN in Appendix Alg. 2).

3 Experiments

To assess NPE-PFN, we conduct experiments on synthetic SBI benchmark tasks and real data, covering scenarios from low to high-dimensional data and including cases with model misspecification.

3.1 Amortized and sequential benchmark performance

We evaluate NPE-PFN on various tasks from the SBI benchmark [27], which provides ground truth posterior samples for 10 observations for each task. We measure posterior sample quality using the classifier two-sample test (C2ST, 53). A C2ST accuracy of 0.5 indicates that the approximate posterior exactly matches the ground truth posterior, as the classifier fails to distinguish between the two sample sets. In contrast, an accuracy of 1.0 reflects a strong mismatch between the estimated and true posterior. As baselines, we compare against flow-based NPE and NLE provided by the SBI library [54, 55]. For sequential baselines, we use truncated sequential NPE (TSNPE [37]) and sequential NLE (SNLE [11]). Posterior estimates are computed across four simulation budgets, ranging from 10^2 to 10^5 simulations. Beyond 10^4 simulations, NPE-PFN incorporates filtering, whereas an alternative unfiltered variant, denoted *NPE-PFN (unfiltered)*, extends the context size beyond the TabPFN-recommended limit. For budgets $\leq 10^4$, both variants are identical.

Amortized inference. For 10^2 simulations, NPE-PFN substantially outperforms NLE and NPE on all but one task (Fig. 2a,b). For 10^3 or 10^4 simulations, NPE-PFN is generally competitive, with

the exception of a few tasks where some baselines outperform it. For example, NLE outperforms NPE-PFN on the simple-likelihood-complex-posterior (SLCP) task, as NPE-PFN, like NPE, directly targets the posterior instead of the simpler likelihood. However, NPE-PFN performs substantially better than both baselines on tasks such as SIR or Lotka-Volterra for all simulation budgets. With 10^5 simulations, the difference between NPE-PFN and its unfiltered variant becomes evident. The extended context does not improve performance for larger simulation budgets, while the filtered NPE-PFN variant achieves significantly better C2ST accuracies, matching or surpassing the baselines. While NPE-PFN is training-free, inference speed depends on the number of simulations, and the dimensionality of the parameter and observation spaces. For the benchmark tasks, we observe an inference speed that is comparable to NLE (Fig. 2b), but orders of magnitude slower than NPE, which is near-instantaneous. We perform a careful analysis of the inference speed of NPE-PFN across all relevant variables for users to make an informed decision in their respective application (Appendix Sec. D.3, Tab. D-2).

The advantage of NPE-PFN over baseline methods remains for other tasks outside the SBI benchmark suite (Appendix Sec. D.1, Fig. D-1), as well as compared to other baseline methods such as NRE, NPE ensembles (Appendix Sec. D.2, Fig. D-2), and even when performing extensive hyperparameter optimization for NPE (Appendix Fig. D-4). Furthermore, NPE-PFN posterior estimates are well-calibrated (Fig. D-3).

In cases where data dimension d_x outgrows the supported feature dimension and learned summary statistics are required, the end-to-end trained NPE, however, outperforms NPE-PFN with independently pre-trained summary statistics (Appendix Sec. D.4, Fig. D-5). In addition, we ablate the effect of the filter size indicating that the default size is (close to) optimal for all considered tasks (Appendix Sec. D.6, Fig. D-7). Finally, we vary the autoregressive order, in which the parameter dimensions are sampled, and find no noticeable effect compared to the default order (Appendix Sec. D.8).

Sequential inference. We adapt the setup in Deistler et al. [37] to evaluate TSNPE-PFN. Rejection sampling is performed using the fast, approximate ratio-based densities (Sec. 2.4, Appendix Sec. D.5, Fig. D-6 for a comparison between autoregressive and ratio-based density evaluation). We equally divide the simulation budget into 10 rounds, with proposals truncated to the $1 - \varepsilon$ highest density region, where $\varepsilon = 10^{-3}$. TSNPE-PFN excels with small simulation budgets (Fig. 2c). On the SIR task, it reaches close to optimal C2ST accuracy with only 10^2 simulations. At larger budgets, TSNPE-PFN maintains a strong performance, particularly on the Lokta-Volterra task, which is highly challenging for other SBI methods. TSNPE-PFN benefits from NPE-PFN's ability to generate high-quality posterior estimates with few simulations, allowing it to acquire more relevant simulations in early rounds.

Summary. NPE-PFN provides accurate posterior estimates, especially for small simulation budgets. With filtering, NPE-PFN is also competitive for larger simulation budgets. As an in-context learning method, it requires no training on specific simulations and offers inference speeds comparable to other baselines, though runtime scales with the number of simulations and the dimensionality of the parameters and observations. This flexibility makes NPE-PFN applicable to a wide range of setups and simulators. We observe NPE-PFN performs particularly well in tasks with underlying graphical structure and conditional (in-)dependencies, which are similar to the structural causal models used in TabPFN's pre-training. In addition to posterior estimation with NPE-PFN, we evaluate the unconditional high-dimensional density estimation capabilities of TabPFN on UCI datasets [56], showing that it is more data-efficient than a neural spline flow (Appendix Sec. D.9, Fig. D-10).

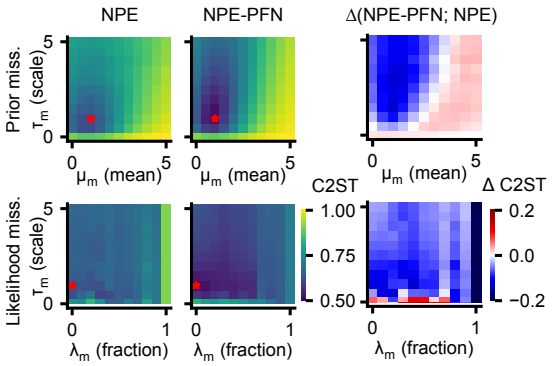


Figure 3: **Robustness under misspecification.** C2ST against a well-specified ground truth posterior. The red star marks the well-specified model. **Rows:** Prior and likelihood misspecification. **Columns:** NPE, NPE-PFN, and their difference in terms of C2ST accuracy (darker blue indicates that NPE-PFN is better).

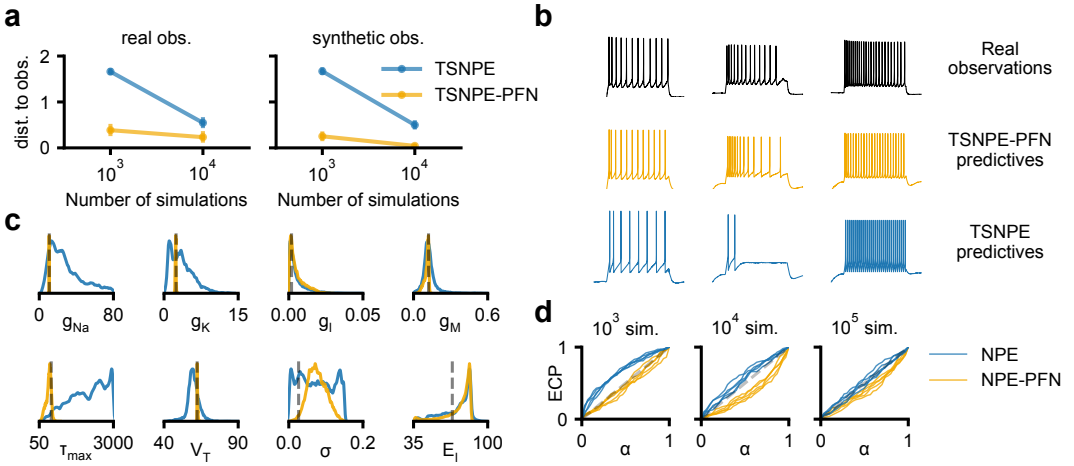


Figure 4: **Posterior inference for observations from the Allen cell type database. (a)** Average distance to observation in standardized space of summary statistics for both the real and synthetic observations. **(b)** Posterior predictive simulations for real observations using TSNPE-PFN and TSNPE. **(c)** Posterior marginals for one synthetic observation; TSNPE-PFN marginals are substantially more constrained for several parameters. **(d)** Simulation-based calibration for NPE-PFN and NPE on the HH simulator.

3.2 Impact of misspecification

To investigate how NPE-PFN handles misspecification, we use the 2D Gaussian means misspecification benchmark from Schmitt et al. [57], considering both prior and likelihood misspecification. The ground truth is given by the prior $\boldsymbol{\mu} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and likelihood $\boldsymbol{x}_i \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{I})$. Prior misspecification uses the prior $\boldsymbol{\mu} \sim \mathcal{N}(\boldsymbol{\mu}_m, \tau_m \boldsymbol{I})$ while keeping the true likelihood; Likelihood misspecification uses the true prior with the likelihood $\boldsymbol{x}_i \sim \lambda_m \text{Beta}(2, 5) + (1 - \lambda_m)\mathcal{N}(\boldsymbol{\mu}, \tau_m \boldsymbol{I})$, where $\tau_m \in \mathbb{R}^+$, $\lambda_m \in [0, 1]$. We evaluate the C2ST of predicted posteriors against the reference posterior distribution of the well-specified data using 10^2 simulations (details in Appendix Sec. 3.2).

Both NPE-PFN and NPE degrade as the level of misspecification increases, with prior misspecification leading to more severe performance degradation than likelihood misspecification (Fig. 3). However, NPE-PFN achieves better C2ST values than NPE across a broader range of misspecification levels. In addition, NPE-PFN remains insensitive to various feature and noise distributions (Appendix Sec. D.7, Fig. D-8). Finally, we find that the performance of NPE-PFN on the SBI benchmark tasks is invariant to the choice of the autoregressive order (Appendix Sec. D.8, Fig. D-9).

Thus, the trend observed in the SBI benchmark extends to these simple but misspecified tasks, on which NPE-PFN also consistently outperforms NPE. Next, we evaluated whether this performance gap also holds for scientific tasks with real data, on which the simulator is potentially misspecified.

3.3 Single-Compartment Hodgkin-Huxley Model

We evaluate the performance of (TS)NPE-PFN on a challenging inference task, considering the Hodgkin-Huxley (HH) simulator of single-neural voltage dynamics [58, 59]. Following Gonçalves et al. [60], the simulator has eight parameters and seven summary statistics based on the voltage trace. We infer posteriors for 10 real observations from the Allen cell type database [61], for which inference is known to be challenging [62–64]. We evaluate inference quality via posterior predictives and the average Euclidean distance between predictives and real observations in a standardized summary statistics space. In addition, for each real observation, we create an associated synthetic, and therefore well-specified observation, where the ground truth parameter is known, allowing for direct comparison between inferred posteriors and true values.

TSNPE-PFN yields posterior predictives that closely match *real observations* with a simulation budget of 10^4 (Fig. 4a,b). It outperforms TSNPE, which struggles to produce realistic predictives, especially at smaller budgets. For *synthetic observations*, TSNPE-PFN again achieves better predictive accuracy

with fewer simulations, with the average distance approaching zero for a simulation budget of 10^4. In parameter space, TSNPE-PFN posteriors are tightly concentrated around the true parameters, with significant improvement over TSNPE (Fig. 4c). In the *amortized setting*, NPE-PFN shows good calibration for 10^3 and 10^5 simulations but is slightly overconfident for 10^4 (Fig. 4d). However, (TS)NPE-PFN achieves significantly better prediction quality and yields tight posteriors around the true parameters, making it more sensitive to miscalibration.

These results demonstrate TSNPE-PFN's superior simulation efficiency, achieving the same predictive quality as TSNPE with an order of magnitude fewer simulations. This efficiency makes TSNPE-PFN particularly well-suited for inference with real, potentially misspecified simulators under limited simulation budgets.

3.4 Pyloric Network Model

Finally, we evaluate TSNPE-PFN on a high-dimensional simulator of the pyloric network in the stomatogastric ganglion (STG) of the crab *Cancer Borealis*, a well-studied circuit generating rhythmic activity [60, 65, 66]. This model consists of three neurons, each with eight membrane conductances, interconnected by seven synapses, resulting in a 31-dimensional parameter space. The simulated voltage traces are summarized using 15 established statistics. A key challenge in this setting is the extreme sparsity of valid simulations. Specifically, 99% of parameter samples from the prior yield implausible voltage traces, preventing the computation of summary statistics [67].

For this reason, earlier work relied on millions of simulations for an amortized NPE-based posterior approximation (18 million in Gonçalves et al. [60]). The simulation count could later be reduced by restricting the prior to *valid* simulation using an additional classifier [67]. In addition, several sequential algorithms have been developed to address this problem [18, 37, 68], the latest of which achieves good posterior predictives using $1.5 \cdot 10^5$ simulations.

To perform sequential inference on the pyloric network with TSNPE-PFN, we use a restricted prior based on the TabPFN classifier. Due to the high dimensionality of the parameter space, rejection sampling is prohibitive. Therefore, we use sampling importance resampling with an oversampling factor of 10. In the first round, we use $5 \cdot 10^3$ simulations and run TSNPE-PFN for 45 rounds, adding 10^3 simulations per round for a total of $5 \cdot 10^4$ simulations (details in Appendix Sec. C.4).

The generated voltage traces from posterior mean estimates of TSNPE-PFN closely resemble true measurements (Fig. 5). With more than 50% valid simulations at just $5 \cdot 10^3$ simulations and approximately 99% validity for fewer than $5 \cdot 10^4$ simulations, TSNPE-PFN outperforms all comparison methods, which either required substantially larger simulation budgets to reach comparable performance or did not reach it at all. Furthermore, the autoregressive evaluation order again has no noticeable effect on the resulting performance (Appendix Sec. C.4).

Our results demonstrate that TSNPE-PFN can infer posteriors with substantially improved simulation efficiency, making it a powerful approach for complex high-dimensional models with a limited simulation budget.

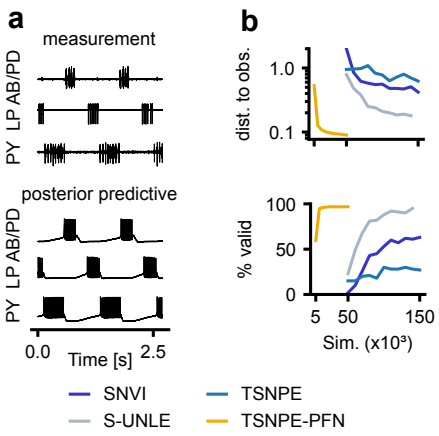


Figure 5: **Results on the pyloric simulator. (a)** Voltage traces from the experimental measurement (top) and a posterior predictive simulated using the posterior mean from TSNPE-PFN as the parameter (bottom). **(b)** Average distance (energy scoring rule) to observation and percentage of *valid* simulation from posterior samples; compared to experimental results obtained in Glaser et al. [68].

4 Discussion

We have shown here that pre-trained foundation models—specifically, TabPFN—can be effectively repurposed to perform accurate training-free simulation-based inference. Together with autoregressive

sampling for high-dimensional posterior distributions, filtering to increase context size, and fast density evaluation using the density ratio, NPE-PFN provides a flexible inference method that can be effortlessly applied to many different inference tasks. Across several benchmarks and two real-world tasks, NPE-PFN consistently estimates accurate posteriors with particularly strong results for small simulation budgets, often requiring orders of magnitude fewer simulations than existing SBI methods. These results establish foundation models and in-context learning as a promising direction for training-free inference in scientific applications.

Related work. Our work relates to **meta-learning** [69], where models generalize across tasks by acquiring shared inductive biases. Meta learning has been applied to amortized (Bayesian) inference [70, 71]. More specifically, however, NPE-PFN is an instance of **in-context learning** (ICL), where task adaptation is achieved solely by conditioning pre-trained models on different inputs [43]. ICL was popularized by LLMs [72], but its scope has expanded beyond language, with work showing that transformers can learn various function classes in context [73, 74] and work introducing prior fitted networks [38, 44, 45]. NPE-PFN extends this direction by enabling ICL for simulation-based Bayesian inference. We note that this does not require training the inference networks on *any* SBI tasks, but rather works well using an "off-the-shelf" model trained for classification and regression.

In addition to NPE-PFN, multiple methods have been proposed to perform **ICL for Bayesian inference** [75–77]. These approaches parameterize posteriors using, for example, Gaussians, flows, or mixtures and require fixed input and parameter dimensionality and, to date, are only trained on a specific, limited class of probabilistic models. In contrast, NPE-PFN leverages autoregressive modeling over parameter dimensions, enabling zero-shot generalization without retraining across inference problems with varying shapes.

Finally, several works have proposed **extensions for TabPFN** beyond its original setting, targeting new data modalities, larger context sizes, or higher-dimensional features. To address TabPFN's limited context size, retrieval-based approaches select informative subsets of the training data [47–49, 78], including nearest-neighbor-based filtering [47–49], which we adapt in NPE-PFN for Bayesian inference. Crucially, we demonstrate that this filtering does not bias the posterior, as the filter step can be interpreted as a calibration kernel [5]. Other strategies include data distillation [79, 80], boosting weak learners [81], feature-wise ensembling [82], encoders and bagging [51], as well as non-autoregressive extensions to time series forecasting [83] and tabular data generation [84]. TabICL [85] proposes architectural changes to scale to datasets with over 10^5 points. NPE-PFN departs from these approaches by repurposing TabPFN for posterior inference through filtering and fast evaluation of densities.

Limitations. While NPE-PFN eliminates the need for simulation-specific training, it inherits properties of TabPFN. TabPFN has a soft limit in maximum feature size (≤ 500) and context length ($\leq 10^4$). We present methods to overcome these limitations, i.e, filtering and embedding nets. However, as the number of simulations and the need for density evaluations increase, the performance advantage of NPE-PFN over standard methods such as NPE diminishes. Furthermore, while embedding networks allow NPE-PFN to be used for high-dimensional observations, they require separate pre-training. In addition, modeling high-dimensional parameter spaces beyond TabPFN's maximum feature size of 500 dimensions remains challenging. Finally, like other in-context methods, NPE-PFN is more expensive than flow-based NPE, a challenge exacerbated by the autoregressive estimation of posteriors. However, unlike other in-context methods for Bayesian inference, NPE-PFN does not require training and hyperparameter tuning, making it suitable for exploratory workflows or non-expert users. In addition, by requiring substantially fewer simulations, NPE-PFN accelerates the overall SBI workflow. Thus, while we expect NPE-PFN to be particularly useful for exploratory workflows or expensive simulators, inference methods based on trained density estimators might still be advantageous in settings where large numbers of simulations are available, or where bespoke embedding networks and fast sampling are required, such as in some applications in astrophysics [22]. An interesting direction for future work is to transfer NPE-PFN's advantages to existing, faster inference techniques (e.g., via student–teacher distillation).

Conclusion. We presented NPE-PFN, a method that repurposes the tabular foundation model TabPFN for training-free simulation-based Bayesian inference. NPE-PFN is competitive with—and often outperforms—existing methods, especially for small simulation budgets. NPE-PFN is a simple-to-use and simulation-efficient tool for SBI, with the potential to become the go-to method for novice users, exploratory workflows, and applications with constrained simulation budgets.

Acknowledgments and Disclosure of Funding

We thank Michael Deistler and Katharina Eggensperger for feedback on the manuscript, specifically Michael Deistler for suggesting to filter simulations and Jonas Beck for the implementation of the pyloric simulator. We thank all members of the Mackelab for discussions and feedback on the manuscript. This work was funded by the German Research Foundation (DFG) under Germany's Excellence Strategy – EXC number 2064/1 – 390727645 and SFB 1233 'Robust Vision' (276693517), the European Union (ERC, DeepCoMechTome, 101089288), the "Certification and Foundations of Safe Machine Learning Systems in Healthcare" project funded by the Carl Zeiss Foundation and the Boehringer Ingelheim AI & Data Science Fellowship Program. JV and MG are members of the International Max Planck Research School for Intelligent Systems (IMPRS-IS).

References

[1] Eric Winsberg. *Science in the age of computer simulation*. University of Chicago Press, 2019.

[2] Benjamin Skuse. The third pillar. *Physics World*, 32(3):40–43, 2019.

[3] Alexander Lavin, David Krakauer, Hector Zenil, Justin Gottschlich, Tim Mattson, Johann Brehmer, Anima Anandkumar, Sanjay Choudry, Kamil Rocki, Atılım Güneş Baydin, Carina Prunkl, Brooks Paige, Olexandr Isayev, Erik Peterson, Peter L. McMahon, Jakob Macke, Kyle Cranmer, Jiaxin Zhang, Haruko Wainwright, Adi Hanuka, Manuela Veloso, Samuel Assefa, Stephan Zheng, and Avi Pfeffer. Simulation intelligence: Towards a new generation of scientific methods. *arXiv preprint arXiv:2112.03235*, 2022.

[4] George Papamakarios and Iain Murray. Fast ε-free inference of simulation models with Bayesian conditional density estimation. *Advances in neural information processing systems*, 29, 2016.

[5] 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.

[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] Maximilian Dax, Stephen R Green, Jonathan Gair, Michael Deistler, Bernhard Schölkopf, and Jakob H. Macke. Group equivariant neural posterior estimation. In *International Conference on Learning Representations*, 2022.

[8] Tomas Geffner, George Papamakarios, and Andriy Mnih. Compositional score modeling for simulation-based inference. In *International Conference on Machine Learning*, pages 11098–11116. PMLR, 2023.

[9] Louis Sharrock, Jack Simons, Song Liu, and Mark Beaumont. Sequential neural score estimation: Likelihood-free inference with conditional score based diffusion models. In *International Conference on Machine Learning*, pages 44565–44602. PMLR, 2024.

[10] Jonas Wildberger, Maximilian Dax, Simon Buchholz, Stephen Green, Jakob H Macke, and Bernhard Schölkopf. Flow matching for scalable simulation-based inference. In *Advances in Neural Information Processing Systems*, volume 36, pages 16837–16864, 2023.

[11] George Papamakarios, David Sterratt, and Iain Murray. Sequential neural likelihood: Fast likelihood-free inference with autoregressive flows. In *International Conference on Artificial Intelligence and Statistics*, pages 837–848. PMLR, 2019.

[12] Jan-Matthis Lueckmann, Giacomo Bassetto, Theofanis Karaletsos, and Jakob H Macke. Likelihood-free inference with emulator networks. In *Symposium on Advances in Approximate Bayesian Inference*, pages 32–53. PMLR, 2019.

[13] Jan Boelts, Jan-Matthis Lueckmann, Richard Gao, and Jakob H Macke. Flexible and efficient simulation-based inference for models of decision-making. *Elife*, 11:e77220, 2022.

[14] Conor Durkan, Iain Murray, and George Papamakarios. On contrastive learning for likelihood-free inference. In *International Conference on Machine Learning*, pages 2771–2781. PMLR, 2020.

[15] Joeri Hermans, Volodimir Begy, and Gilles Louppe. Likelihood-free MCMC with amortized approximate ratio estimators. In *International Conference on Machine Learning*, pages 4239–4248. PMLR, 2020.

[16] Joeri Hermans, Arnaud Delaunoy, François Rozet, Antoine Wehenkel, Volodimir Begy, and Gilles Louppe. A crisis in simulation-based inference? Beware, your posterior approximations can be unfaithful. *Transactions on Machine Learning Research*, 2022.

[17] Benjamin K Miller, Christoph Weniger, and Patrick Forré. Contrastive neural ratio estimation. In *Advances in Neural Information Processing Systems*, volume 35, pages 3262–3278, 2022.

[18] Manuel Gloeckler, Michael Deistler, and Jakob H. Macke. Variational methods for simulation-based inference. In *International Conference on Learning Representations*, 2022.

[19] Stefan T. Radev, Marvin Schmitt, Valentin Pratz, Umberto Picchini, Ullrich Köthe, and Paul-Christian Bürkner. Jana: Jointly amortized neural approximation of complex Bayesian models. In *Conference on Uncertainty in Artificial Intelligence*, volume 216, pages 1695–1706. PMLR, 2023.

[20] He Jia. Simulation-based inference with quantile regression. *International Conference on Machine Learning*, 2024.

[21] Manuel Gloeckler, Michael Deistler, Christian Dietrich Weilbach, Frank Wood, and Jakob H. Macke. All-in-one simulation-based inference. In *International Conference on Machine Learning*, volume 235, pages 15735–15766. PMLR, 2024.

[22] Maximilian Dax, Stephen R Green, Jonathan Gair, Nihar Gupte, Michael Pürrer, Vivien Raymond, Jonas Wildberger, Jakob H Macke, Alessandra Buonanno, and Bernhard Schölkopf. Real-time inference for binary neutron star mergers using machine learning. *Nature*, 639(8053): 49–53, 2025.

[23] Lukas N. Groschner, Jonatan G. Malis, Birte Zuidinga, and Alexander Borst. A biophysical account of multiplication by a single neuron. *Nature*, 603(7899):119–123, 2022.

[24] Mark A Beaumont, Wenyang Zhang, and David J Balding. Approximate Bayesian computation in population genetics. *Genetics*, 162(4):2025–2035, 2002.

[25] Mark A Beaumont, Jean-Marie Cornuet, Jean-Michel Marin, and Christian P Robert. Adaptive approximate Bayesian computation. *Biometrika*, 96(4):983–990, 2009.

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

[27] Jan-Matthis Lueckmann, Jan Boelts, David Greenberg, Pedro Goncalves, and Jakob Macke. Benchmarking simulation-based inference. In *International Conference on Artificial Intelligence and Statistics*, pages 343–351. PMLR, 2021.

[28] Atilim Güneş Baydin, Lei Shao, Wahid Bhimji, Lukas Heinrich, Lawrence Meadows, Jialin Liu, Andreas Munk, Saeid Naderiparizi, Bradley Gram-Hansen, Gilles Louppe, et al. Etalumis: Bringing probabilistic programming to scientific simulators at scale. In *Proceedings of the international conference for high performance computing, networking, storage and analysis*, pages 1–24, 2019.

[29] 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.

[30] Christian Weilbach, Boyan Beronov, Frank Wood, and William Harvey. Structured conditional continuous normalizing flows for efficient amortized inference in graphical models. In *International Conference on Artificial Intelligence and Statistics*, pages 4441–4451. PMLR, 2020.

[31] Manuel Gloeckler, Shoji Toyota, Kenji Fukumizu, and Jakob H. Macke. Compositional simulation-based inference for time series. In *International Conference on Learning Representations*, 2025.

[32] Anastasia N Krouglova, Hayden R Johnson, Basile Confavreux, Michael Deistler, and Pedro J Gonçalves. Multifidelity simulation-based inference for computationally expensive simulators. *arXiv preprint arXiv:2502.08416*, 2025.

[33] Caroline Tatsuoka, Minglei Yang, Dongbin Xiu, and Guannan Zhang. Multi-fidelity parameter estimation using conditional diffusion models. *arXiv preprint arXiv:2504.01894*, 2025.

[34] Fredrik Wrede, Robin Eriksson, Richard Jiang, Linda Petzold, Stefan Engblom, Andreas Hellander, and Prashant Singh. Robust and integrative Bayesian neural networks for likelihood-free parameter inference. In *International Joint Conference on Neural Networks*, pages 1–10. IEEE, 2022.

[35] Pablo Lemos, Miles Cranmer, Muntazir Abidi, ChangHoon Hahn, Michael Eickenberg, Elena Massara, David Yallup, and Shirley Ho. Robust simulation-based inference in cosmology with Bayesian neural networks. *Machine Learning: Science and Technology*, 4(1):01LT01, 2023.

[36] Arnaud Delaunoy, Maxence de la Brassinne Bonardeaux, Siddharth Mishra-Sharma, and Gilles Louppe. Low-budget simulation-based inference with Bayesian neural networks. *arXiv preprint arXiv:2408.15136*, 2024.

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

[38] Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister, and Frank Hutter. Accurate predictions on small data with a tabular foundation model. *Nature*, 637(8045):319–326, 2025.

[39] Tuan Anh Le, Atilim Giineş Baydin, Robert Zinkov, and Frank Wood. Using synthetic data to train neural networks is model-based reasoning. In *International Joint Conference on Neural Networks (IJCNN)*, pages 3514–3521. IEEE, 2017.

[40] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International Conference on Machine Learning*, volume 37, pages 1530–1538. PMLR, 2015.

[41] George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

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

[43] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Baobao Chang, Xu Sun, Lei Li, and Zhifang Sui. A survey on in-context learning. In *Conference on Empirical Methods in Natural Language Processing*, pages 1107–1128, 2024.

[44] Samuel Müller, Noah Hollmann, Sebastian Pineda Arango, Josif Grabocka, and Frank Hutter. Transformers can do Bayesian inference. In *International Conference on Learning Representations*, 2022.

[45] Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. TabPFN: A transformer that solves small tabular classification problems in a second. In *International Conference on Learning Representations*, 2023.

[46] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[47] Thomas Nagler. Statistical foundations of prior-data fitted networks. In *International Conference on Machine Learning*, pages 25660–25676. PMLR, 2023.

[48] Valentin Thomas, Junwei Ma, Rasa Hosseinzadeh, Keyvan Golestan, Guangwei Yu, Maks Volkovs, and Anthony L Caterini. Retrieval & fine-tuning for in-context tabular models. *Advances in Neural Information Processing Systems*, 37:108439–108467, 2024.

[49] Mykhailo Koshil, Thomas Nagler, Matthias Feurer, and Katharina Eggensperger. Towards localization via data embedding for TabPFN. In *NeurIPS 2024 Third Table Representation Learning Workshop*, 2024.

[50] Yanzhi Chen, Dinghuai Zhang, Michael U. Gutmann, Aaron Courville, and Zhanxing Zhu. Neural approximate sufficient statistics for implicit models. In *International Conference on Learning Representations*, 2021.

[51] Si-Yang Liu and Han-Jia Ye. TabPFN unleashed: A scalable and effective solution to tabular classification problems. *arXiv preprint arXiv:2502.02527*, 2025.

[52] Masashi Sugiyama, Takafumi Kanamori, Taiji Suzuki, Shohei Hido, Jun Sese, Ichiro Takeuchi, and Liwei Wang. A density-ratio framework for statistical data processing. *IPSJ Transactions on Computer Vision and Applications*, 1:183–208, 2009.

[53] David Lopez-Paz and Maxime Oquab. Revisiting classifier two-sample tests. In *International Conference on Learning Representations*, 2017.

[54] 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. *Journal of Open Source Software*, 5(52):2505, 2020.

[55] Jan Boelts, Michael Deistler, Manuel Gloeckler, Álvaro Tejero-Cantero, Jan-Matthis Lueckmann, Guy Moss, Peter Steinbach, Thomas Moreau, Fabio Muratore, Julia Linhart, Conor Durkan, Julius Vetter, Benjamin Kurt Miller, Maternus Herold, Abolfazl Ziaeemehr, Matthijs Pals, Theo Gruner, Sebastian Bischoff, Nastya Krouglova, Richard Gao, Janne K. Lappalainen, Bálint Mucsányi, Felix Pei, Auguste Schulz, Zinovia Stefanidi, Pedro Rodrigues, Cornelius Schröder, Faried Abu Zaid, Jonas Beck, Jaivardhan Kapoor, David S. Greenberg, Pedro J. Gonçalves, and Jakob H. Macke. sbi reloaded: a toolkit for simulation-based inference workflows. *Journal of Open Source Software*, 10(108):7754, 2025.

[56] Andrew Frank. UCI machine learning repository. *http://archive. ics. uci. edu/ml*, 2010.

[57] Marvin Schmitt, Paul-Christian Bürkner, Ullrich Köthe, and Stefan T Radev. Detecting model misspecification in amortized bayesian inference with neural networks: An extended investigation. *arXiv preprint arXiv:2406.03154*, 2024.

[58] A. L. Hodgkin and A. F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *J Physiol*, 117(4):500–544, 1952. doi: 10.1113/jphysiol.1952.sp004764.

[59] Martin Pospischil, Maria Toledo-Rodriguez, Cyril Monier, Zuzanna Piwkowska, Thierry Bal, Yves Frégnac, Henry Markram, and Alain Destexhe. Minimal Hodgkin–Huxley type models for different classes of cortical and thalamic neurons. *Biological cybernetics*, 99:427–441, 2008.

[60] 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.

[61] Allen Institute for Brain Science. Allen cell types database. `https://celltypes.brain-map.org`, 2015.

[62] Richard Gao, Michael Deistler, and Jakob H Macke. Generalized Bayesian inference for scientific simulators via amortized cost estimation. *Advances in Neural Information Processing Systems*, 36:80191–80219, 2023.

[63] Nicholas Tolley, Pedro LC Rodrigues, Alexandre Gramfort, and Stephanie R Jones. Methods and considerations for estimating parameters in biophysically detailed neural models with simulation based inference. *PLOS Computational Biology*, 20(2):e1011108, 2024.

[64] Yves Bernaerts, Michael Deistler, Pedro J Goncalves, Jonas Beck, Marcel Stimberg, Federico Scala, Andreas S Tolias, Jakob H Macke, Dmitry Kobak, and Philipp Berens. Combined statistical-mechanistic modeling links ion channel genes to physiology of cortical neuron types. *bioRxiv*, 2023.

[65] Astrid A Prinz, Cyrus P Billimoria, and Eve Marder. Alternative to hand-tuning conductance-based models: construction and analysis of databases of model neurons. *Journal of neurophysiology*, 2003.

[66] Astrid A Prinz, Dirk Bucher, and Eve Marder. Similar network activity from disparate circuit parameters. *Nature neuroscience*, 7(12):1345–1352, 2004.

[67] Michael Deistler, Jakob H Macke, and Pedro J Gonçalves. Energy-efficient network activity from disparate circuit parameters. *Proceedings of the National Academy of Sciences*, 119(44): e2207632119, 2022.

[68] Pierre Glaser, Michael Arbel, Samo Hromadka, Arnaud Doucet, and Arthur Gretton. Maximum likelihood learning of unnormalized models for simulation-based inference. *arXiv preprint arXiv:2210.14756*, 2023.

[69] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44 (9):5149–5169, 2021.

[70] Mike Wu, Kristy Choi, Noah Goodman, and Stefano Ermon. Meta-amortized variational inference and learning. In *Conference on Artificial Intelligence*, volume 34, pages 6404–6412, 2020.

[71] Ekaterina Iakovleva, Jakob Verbeek, and Karteek Alahari. Meta-learning with shared amortized variational inference. In *International Conference on Machine Learning*, pages 4572–4582. PMLR, 2020.

[72] 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.

[73] Shivam Garg, Dimitris Tsipras, Percy S Liang, and Gregory Valiant. What can transformers learn in-context? A case study of simple function classes. *Advances in Neural Information Processing Systems*, 35:30583–30598, 2022.

[74] Madhur Panwar, Kabir Ahuja, and Navin Goyal. In-context learning through the Bayesian prism. In *International Conference on Learning Representations*, 2024.

[75] Sarthak Mittal, Niels Leif Bracher, Guillaume Lajoie, Priyank Jaini, and Marcus Brubaker. Amortized in-context bayesian posterior estimation. *arXiv preprint arXiv:2502.06601*, 2025.

[76] Arik Reuter, Tim GJ Rudner, Vincent Fortuin, and David Rügamer. Can transformers learn full Bayesian inference in context? *arXiv preprint arXiv:2501.16825*, 2025.

[77] George Whittle, Juliusz Ziomek, Jacob Rawling, and Michael A Osborne. Distribution transformers: Fast approximate Bayesian inference with on-the-fly prior adaptation. *arXiv preprint arXiv:2502.02463*, 2025.

[78] Derek Qiang Xu, F Olcay Cirit, Reza Asadi, Yizhou Sun, and Wei Wang. Mixture of in-context prompters for tabular PFNs. In *International Conference on Learning Representations*, 2025.

[79] Junwei Ma, Valentin Thomas, Guangwei Yu, and Anthony L. Caterini. In-context data distillation with TabPFN. In *ICLR 2024 Workshop on Mathematical and Empirical Understanding of Foundation Models*, 2024.

[80] Benjamin Feuer, Robin Schirrmeister, Valeriia Cherepanova, Chinmay Hegde, Frank Hutter, Micah Goldblum, Niv Cohen, and Colin White. Tunetables: Context optimization for scalable prior-data fitted networks. *Advances in Neural Information Processing Systems*, 37:83430–83464, 2024.

[81] Yuxin Wang, Botian Jiang, YiranGuo, Quan Gan, David Wipf, Xuanjing Huang, and Xipeng Qiu. Prior-fitted networks scale to larger datasets when treated as weak learners. In *International Conference on Artificial Intelligence and Statistics*, 2025.

[82] Han-Jia Ye, Si-Yang Liu, and Wei-Lun Chao. A closer look at TabPFN v2: Strength, limitation, and extension. *arXiv preprint arXiv:2502.17361*, 2025.

[83] Shi Bin Hoo, Samuel Müller, David Salinas, and Frank Hutter. The tabular foundation model TabPFN outperforms specialized time series forecasting models based on simple features. In *NeurIPS 2024 Third Table Representation Learning Workshop*, 2024.

[84] Junwei Ma, Apoorv Dankar, George Stein, Guangwei Yu, and Anthony Caterini. TabPFGen – tabular data generation with TabPFN. In *NeurIPS 2023 Second Table Representation Learning Workshop*, 2023.

[85] Jingang Qu, David Holzmüller, Gaël Varoquaux, and Marine Le Morvan. TabICL: A tabular foundation model for in-context learning on large data. *arXiv preprint arXiv:2502.05564*, 2025.

[86] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

[87] Omry Yadan. Hydra - a framework for elegantly configuring complex applications. Github, 2019.

[88] Benjamin Feuer, Niv Cohen, and Chinmay Hegde. Scaling TabPFN: Sketching and feature selection for tabular prior-data fitted networks. In *NeurIPS 2023 Second Table Representation Learning Workshop*, 2023.

[89] Tianhong Li, Dina Katabi, and Kaiming He. Return of unconditional generation: A self-supervised representation generation method. *Advances in Neural Information Processing Systems*, 37:125441–125468, 2024.

[90] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

[91] Michael Deistler, Kyra L. Kadhim, Matthijs Pals, Jonas Beck, Ziwei Huang, Manuel Gloeckler, Janne K. Lappalainen, Cornelius Schröder, Philipp Berens, Pedro J. Gonçalves, and Jakob H. Macke. Differentiable simulation enables large-scale training of detailed biophysical models of neural dynamics. *bioRxiv*, pages 2024–08, 2024.

[92] Christian Dietrich Weilbach, William Harvey, and Frank Wood. Graphically structured diffusion models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 36887–36909. PMLR, 2023.

[93] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[94] Manuel Gloeckler, Michael Deistler, and Jakob H. Macke. Adversarial robustness of amortized Bayesian inference. In *International Conference on Machine Learning*, volume 202, pages 11493–11524. PMLR, 2023.

[95] Pablo Lemos, Adam Coogan, Yashar Hezaveh, and Laurence Perreault-Levasseur. Sampling-based accuracy testing of posterior estimators for general inference. In *International Conference on Machine Learning*, pages 19256–19273. PMLR, 2023.

[96] In-Kwon Yeo and Richard A Johnson. A new family of power transformations to improve normality or symmetry. *Biometrika*, 87(4):954–959, 2000.

[97] Will Grathwohl, Ricky T. Q. Chen, Jesse Bettencourt, and David Duvenaud. Scalable reversible generative models with free-form continuous dynamics. In *International Conference on Learning Representations*, 2019.

Appendix Table of Contents

A Software and computational resources

Our implementation of NPE-PFN is based on PyTorch [86] and the TabPFN [38] library. We use the SBI library [54] to run the SBI baselines, and hydra [87] for experiment and hyperparameter management. Code to use NPE-PFN and reproduce the results is available at `https://github.com/mackelab/npe-pfn`.

We use a mix of Nvidia 2080TI, A100, and H100 GPUs to obtain the results related to NPE-PFN. Some results, such as those for the unfiltered NPE-PFN variant together with a simulation budget of 10^5 require the use of GPUs with high VRAM due to the large context used. SBI baselines were run on 8 CPU cores, as normalizing flow training and sampling only benefit from GPUs for large sample sizes and dimensionalities.

B Additional method details

B.1 Pseudocode: NPE-PFN and TSNPE-PFN

In Alg. 1 and Alg. 2, we provide pseudocode for NPE-PFN and TSNPE-PFN, respectively. Note that NPE-PFN is training-free. In practice, this means that training consists of storing the training data $\mathcal{D} = \{(\boldsymbol{\theta}_i, \boldsymbol{x}_i)\}_{i=1}^N$ for in-context processing during inference.

Algorithm 1 NPE-PFN

Require: Observation $\boldsymbol{x}_o$, simulations $\mathcal{D}_{\text{full}} = \{(\boldsymbol{\theta}_i, \boldsymbol{x}_i)\}_{i=1}^N$, TabPFN regressor $q_\psi^{\text{reg}}(\cdot \mid \cdot)$, filter size N_{filter}

if $N_{\text{filter}} < N$ **then** ▷ if filtering is active
  $d_i = d(\boldsymbol{x}_o, \boldsymbol{x}_i)$ ▷ compute filter distances
  $\mathcal{D} \leftarrow \{(\boldsymbol{\theta}_i, \boldsymbol{x}_i) : i \in \text{argtop}_{N_{\text{filter}}}(d_i)\}$ ▷ select N_{filter} closest simulations
else
  $\mathcal{D} \leftarrow \mathcal{D}_{\text{full}}$
end if
for $j = 1, \dots, d_{\boldsymbol{\theta}}$ **do** ▷ for all parameter dimension $d_{\boldsymbol{\theta}}$
  $\mathcal{D}^{<j} = \{\theta_i^j, [\boldsymbol{\theta}_i^{<j}, \boldsymbol{x}_i]\}$ ▷ build context data set
  $\theta^j \sim q_\psi^{\text{reg}}(\theta^j \mid \boldsymbol{\theta}^{<j}, \boldsymbol{x}_o, \mathcal{D}^{<j})$ ▷ sample one parameter dimension from TabPFN
end for
return $\boldsymbol{\theta} = [\theta_1, \theta_2, \dots, \theta_{d_\theta}]^\top$ ▷ posterior sample

B.2 Filtering to increase effective context size

To allow NPE-PFN to make use of more than 10^4 simulations, we filter the simulations based on their relevance to a given observation. In the following, we show that this procedure does not bias the estimation of the posterior and discuss details and possible extensions.

For a given prior $p(\boldsymbol{\theta})$ and likelihood $p(\boldsymbol{x} \mid \boldsymbol{\theta})$, the joint distribution is given by $p(\boldsymbol{x}, \boldsymbol{\theta}) = p(\boldsymbol{x} \mid \boldsymbol{\theta})p(\boldsymbol{\theta})$ and the posterior distribution is given by

$$p(\boldsymbol{\theta} \mid \boldsymbol{x}) = \frac{p(\boldsymbol{x} \mid \boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\boldsymbol{x})} = \frac{p(\boldsymbol{x}, \boldsymbol{\theta})}{\int p(\boldsymbol{x}, \boldsymbol{\theta}')\, d\boldsymbol{\theta}'}. \tag{B-1}$$

Formally, a filter is a non-negative function $f_{\boldsymbol{x}_o}(\boldsymbol{x})$ based on some observation $\boldsymbol{x}_o$, where we require $f_{\boldsymbol{x}_o}(\boldsymbol{x}_o) > 0$. Crucially, reweighing the joint distribution by the filter does not affect the posterior. More specifically, with the reweighted joint $\tilde{p}(\boldsymbol{x}, \boldsymbol{\theta}) \propto p(\boldsymbol{x}, \boldsymbol{\theta}) f_{\boldsymbol{x}_o}(\boldsymbol{x})$, we have

$$\tilde{p}(\boldsymbol{\theta} \mid \boldsymbol{x}) = \frac{\tilde{p}(\boldsymbol{x}, \boldsymbol{\theta})}{\int \tilde{p}(\boldsymbol{x}, \boldsymbol{\theta}')\, d\boldsymbol{\theta}'} = \frac{f_{\boldsymbol{x}_o}(\boldsymbol{x}) p(\boldsymbol{x}, \boldsymbol{\theta})}{\int f_{\boldsymbol{x}_o}(\boldsymbol{x}) p(\boldsymbol{x}, \boldsymbol{\theta}')\, d\boldsymbol{\theta}'} = p(\boldsymbol{\theta} \mid \boldsymbol{x}), \tag{B-2}$$

as $f_{\boldsymbol{x}_o}(\boldsymbol{x})$ cancels out due to its independence from $\boldsymbol{\theta}$.

In theory, any function f satisfying these properties can be used to filter for relevant simulations. In this work, we only consider an ABC-like filter, where $f_{\boldsymbol{x}_o}(\boldsymbol{x}) = \mathbb{I}(d(\boldsymbol{x}, \boldsymbol{x}_o) < \epsilon)$ is the indicator

Algorithm 2 TSNPE-PFN with rejection sampling

Require: Observation $\boldsymbol{x}_o$, simulator $p(\boldsymbol{x} \mid \boldsymbol{\theta})$, prior $p(\boldsymbol{\theta})$, TabPFN classifier $q_\psi^{\text{cls}}(\cdot \mid \cdot)$, TabPFN regressor $q_\psi^{\text{reg}}(\cdot \mid \cdot)$, number of rounds R, number of posterior samples for ratio M, simulations per round N_r, truncation fraction α

— Initialize dataset —

Draw N_r prior samples $\boldsymbol{\theta}_i \sim p(\boldsymbol{\theta})$
Simulate samples $\boldsymbol{x}_i \sim p(\boldsymbol{x} \mid \boldsymbol{\theta}_i)$
Initialize $\mathcal{D} = \{(\boldsymbol{\theta}_i, \boldsymbol{x}_i)\}_{i=1}^{N_r}$

for $r = 1, \ldots, R$ **do**

— Construct fast ratio-based density estimator —

Draw M posterior samples $\boldsymbol{\theta}_j^{\text{post}} \sim q(\boldsymbol{\theta} \mid \boldsymbol{x}_o)$ using Alg. 1 with $\mathcal{D}$ as context
Assign class $y = 1$ to $\boldsymbol{\theta}_j^{\text{post}}$ and $y = 0$ to $\boldsymbol{\theta}_j^{\text{uni}} \sim \mathcal{U}(\boldsymbol{\theta}_{\min}, \boldsymbol{\theta}_{\max})$
Define classifier dataset $\mathcal{D}_{\text{cls}} = \{(\boldsymbol{\theta}_j, y_j)\}_{j=1}^{2M}$ combining $\boldsymbol{\theta}_j^{\text{post}}$ and $\boldsymbol{\theta}_j^{\text{uni}}$
With $\mathcal{D}_{\text{cls}}$ as context, q_ψ^{cls} can be used to estimate posterior density:

$$q_{\text{cls}}(\boldsymbol{\theta} \mid \boldsymbol{x}_o) = \frac{q_\psi^{\text{cls}}(y = 1 \mid \boldsymbol{\theta}, \mathcal{D}_{\text{cls}})}{1 - q_\psi^{\text{cls}}(y = 1 \mid \boldsymbol{\theta}, \mathcal{D}_{\text{cls}})}.$$

— Estimate high-density region (HDR) —

qs $= \{q_{\text{cls}}(\boldsymbol{\theta}_j^{\text{post}} | \mathbf{x}_o) \text{ for } j = 1, \ldots, M\}$ ▷ evaluate posterior densities
k_α = quantile(qs, α) ▷ estimate HDR threshold

— Rejection sample from truncated prior —

$D_r = \emptyset$
while $|D_r| < N_r$ **do**
$\boldsymbol{\theta} \sim p(\boldsymbol{\theta})$ ▷ sample from prior
if $q_{\text{cls}}(\boldsymbol{\theta} \mid \mathbf{x}_o) \geq k_\alpha$ **then** ▷ accept if in HDR
$\mathbf{x} \sim p(\mathbf{x} \mid \boldsymbol{\theta})$
$D_r \leftarrow D_r \cup \{(\boldsymbol{\theta}, \mathbf{x})\}$
end if
end while
$\mathcal{D} = \mathcal{D} \cup \mathcal{D}_r$
end for

— Sample final samples —

Draw posterior samples $\boldsymbol{\theta}_i \sim q(\boldsymbol{\theta} \mid \boldsymbol{x}_o)$ using Alg. 1 with $\mathcal{D}$ as context
return $\{\boldsymbol{\theta}_i\}_i$

function that evaluates to 1, when some distance $d(\boldsymbol{x}, \boldsymbol{x}_o)$ between the observation $\boldsymbol{x}_o$ simulation $\boldsymbol{x}$ is smaller than some threshold $\epsilon > 0$, and is 0 otherwise. As described in Sec. 2.3, we use the Euclidean distance in feature-wise standardized observation space and choose ϵ adaptively to include the 10^4 closest simulations.

The autoregressive manner in which we use TabPFN for density estimation allows for more complex filter designs that consider not only the observation $\boldsymbol{x}_o$, but also already processed parameter dimensions. In particular, it is possible to select parameter-simulation pairs $(\boldsymbol{\theta}, \boldsymbol{x})$ based on their relevance to the parameter dimensions $<j$ *and* observation $\boldsymbol{x}_o$, when using TabPFN to estimate $p(\theta^j \mid \boldsymbol{x}_o, \boldsymbol{\theta}^{<j}) = q_\psi(\theta^j \mid \boldsymbol{x}_o, \boldsymbol{\theta}^{<j}, \mathcal{D}^{<j})$. We do not investigate this approach here and leave the design of more complex (autoregressive) filters to future work.

B.3 Embedding networks for high-dimensional data

For NPE, it is typical to employ an embedding net $e = f_\phi(\boldsymbol{x})$ depending on the case of high-dimensional data. The embedding network can be end-to-end trained together with the associated

conditional density estimator $q_\phi(\boldsymbol{\theta} \mid f_\phi(\boldsymbol{x}))$ and helps to identify lower-dimensional "sufficient" statistics, which the density estimator can more effectively handle [5]. The embedding net is usually chosen to follow existing properties of the data at hand; i.e., for image-like simulation, one might employ a convolutional neural network.

In principle, this end-to-end approach is directly applicable to NPE-PFN, but it would require differentiation through the TabPFN model, which can be expensive. We instead aim to obtain lower dimensional embedding, i.e., "summary features" independent from the pre-trained model. This idea is similar to feature selection schemes usually employed for TabPFN [51, 88].

While various approaches exist for addressing this problem, we investigate the method proposed by Chen et al. [50], which focuses on maximizing the mutual information between the parameters $\boldsymbol{\theta}$ and their corresponding embeddings $e = f_\phi(\boldsymbol{x})$, with data $\boldsymbol{x} \sim p(\boldsymbol{x} \mid \boldsymbol{\theta})$. The goal is to derive embeddings that serve as "sufficient" statistics for accurately inferring parameters from observed data. However, directly maximizing mutual information is generally intractable, leading to the development of proxy measures.

We employ the distance-correlation proxy, which aligns the distances between embeddings with the distances between their associated parameters. Specifically, we optimize the following loss function:

$$\mathcal{L}(\phi) = \frac{\mathbb{E}_{p(\boldsymbol{x},\theta)p(\boldsymbol{x}',\theta')}\left[d(\theta,\theta')\, d(f_\phi(\boldsymbol{x}), f_\phi(\boldsymbol{x}'))\right]}{\sqrt{\mathbb{E}_{p(\theta)p(\theta')}[d(\theta,\theta')^2]\, \mathbb{E}_{p(\boldsymbol{x})p(\boldsymbol{x}')}[d(f_\phi(\boldsymbol{x}), f_\phi(\boldsymbol{x}'))^2]}} \tag{B-3}$$

We refer to this approach as NPE-PFN-Infomax and present results on using NPE-PFN together with pre-trained embedding nets in Sec. D.4.

B.4 Unconditional density estimation

As outlined in Sec. 2.1, TabPFN is designed as a *conditional* density estimator for one-dimensional densities. For higher-dimensional conditional density estimation, TabPFN can be applied autoregressively. To adapt TabPFN for unconditional density estimation, random Gaussian noise is added as the first dimension. The subsequent conditional density estimation is then performed in the same way as for conditional tasks [38].

For conditional density estimation (e.g., posterior estimation), filtering provides an effective method to optimize the context given the corresponding condition. However, this strategy does not generalize to unconditional density estimation. One solution is to manually introduce a conditional structure. Specifically, we partition the support of the target density into independent clusters and estimate each component separately. Formally, we express the target density as a mixture model

$$p(\boldsymbol{x}) = \sum_{i=1}^{n} p(\boldsymbol{x} \mid c_i)\, p(c_i), \tag{B-4}$$

where $p(c_i)$ is the probability of sampling cluster c_i, and $p(\boldsymbol{x} \mid c_i)$ is the cluster-specific density estimated via NPE-PFN. This mixture model approach effectively increases the usable context size with the number of clusters. In this work, we apply k-means clustering, which has been explored for TabPFN by Xu et al. [78] outside of density estimation. Other methods to introduce conditional structure, such as those proposed by Li et al. [89], represent promising directions for future research. We present results on unconditional density estimation in Sec. D.9.

C Experimental details

C.1 SBI benchmark simulators for amortized and sequential inference

To benchmark NPE-PFN and the sequential version TSNPE-PFN, we use a set of tasks from the SBI benchmark [27]. This benchmark contains several challenging inference tasks with nonlinear dependencies and multi-modal posteriors. We provide an overview of the dimensionality of parameters and observations for all benchmark tasks (Tab. C-1). As described in Sec. 3.1, we measure the quality of the inferred posteriors with a classifier-two-sample-test (C2ST). For the classifier, we use a random forest with the default hyperparameters provided by the SBI library [54].

Table C-1: **Overview on the SBI benchmark tasks.** For details on the specification of the prior and the exact simulator equations, we refer to Lueckmann et al. [27, Appendix T].

Task Name	$\dim \boldsymbol{\theta}$	$\dim \boldsymbol{x}$	**Notes**
Gaussian Linear	10	10	Mean inference in 10-D Gaussian with fixed covariance
Gaussian Mixture	2	2	Common mean of two 2-D Gaussians with varying covariance
Two Moons	2	2	Bimodal, two moon-shaped posterior structure
SLCP	5	8	Simple Gaussian likelihood, complex posterior
Bernoulli GLM	10	10	GLM with Bernoulli observations, smoothness prior
SIR	2	10	Epidemic SIR model, inference of contact and recovery rates
Lotka-Volterra	4	20	Predator-prey dynamics, species interaction parameters

The baseline methods NPE, NLE, NRE and their sequential variants are implemented using the SBI library. All methods use the default normalizing flows (neural spline flow [42] for NPE, masked autoregressive flow [41] for NLE) with hyperparameters suggested by the SBI library. Training was performed using the Adam optimizer [90] with a batch size of 200 and a learning rate of $5 \cdot 10^{-4}$. Training was stopped early based on the validation loss, as evaluated on a held-out set containing 10% of the available simulations. For the NPE ensembles, five different estimators are trained with different random seeds to obtain a mixture of equally weighted estimators.

In all experiments, we use the default version of the TabPFN classifier or regressor for (TS)NPE-PFN, with no changes to hyperparameters such as the softmax temperature. All runtimes for NPE-PFN (Fig. 2b) were obtained using an Nvidia A100 GPU, where possible. For the unfiltered variant of NPE-PFN, an H100 GPU was used for the large context containing 10^5 simulations.

C.2 Misspecification simulator

Here, we detail the experimental setup of Sec. 3.2. The misspecification benchmark from Schmitt et al. [57] is given by

$$\textbf{Truth} \quad \text{Prior:} \quad \boldsymbol{\mu} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}), \qquad \text{Likeli.:} \quad \boldsymbol{x}_i \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{I}) \tag{C-5}$$

$$\textbf{Prior} \quad \text{Prior:} \quad \boldsymbol{\mu} \sim \mathcal{N}(\boldsymbol{\mu}_m, \tau_m \boldsymbol{I}), \qquad \text{Likeli.:} \quad \boldsymbol{x}_i \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{I}) \tag{C-6}$$

$$\textbf{Likeli.} \quad \text{Prior:} \quad \boldsymbol{\mu} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}), \qquad \text{Likeli.:} \quad \boldsymbol{x}_i \sim \lambda \text{Beta}(2, 5) + (1 - \lambda)\mathcal{N}(\boldsymbol{\mu}, \tau_m \boldsymbol{I}) \tag{C-7}$$

where $\mu_m \in \mathbb{R}$ as mean shift, $\tau_m \in \mathbb{R}^+$ as standard deviation scaling, and $\lambda \in [0, 1]$ as the fraction of how many samples are from the true normal or a different beta distribution.

For the experiments, we ran 100 simulations for three seeds over all combinations of (μ_m, τ_m, λ). NPE and NPE-PFN are trained to predict posterior samples. These samples are compared to the reference posterior distribution using C2ST. This reference posterior distribution is defined with respect to the well-specified distribution. Thus, this task is designed to evaluate the robustness of the model under prior or likelihood misspecification.

C.3 Single-compartment Hodgkin-Huxley simulator

To perform inference on the observations from the Allen cell types database [61], we use a single-compartment Hodkin-Huxley model following the one proposed in Pospischil et al. [59]. This model has four types of conductances (sodium, delayed-rectifier potassium, slow voltage-dependent potassium, and leak) and a total of eight parameters. Following Gonçalves et al. [60], who previously used this model for simulation-based inference, we compute seven summary statistics on the simulated and observed action potentials. These are spike count, mean, and standard deviation of the resting potential, and the first four voltage moments (mean, standard deviation, skew, and kurtosis). We also use the same uniform prior as in Gonçalves et al. [60]. For sequential inference with TSNPE and TSNPE-PFN, we use a total of 10^3 or 10^4 simulations, equally divided over five rounds. For both methods, we sample from the truncated proposal using rejection sampling, taking into account the $1 - \varepsilon$ highest density region with $\varepsilon = 10^{-3}$.

Here, we provide posterior predictive samples for all 10 observations from the Allen cell types database using TSNPE-PFN (Fig. D-11) and TSNPE (Fig. D-12), complementing the results presented in the main text (Fig. 4). Importantly, both inference and the computation of the average predictive

distance are performed on the summary statistics and not directly on the simulations or observations. Thus, features beyond those encoded by the summary statistics are not captured. For TSNPE-PFN, the posterior predictives closely match with the observations in most cases, with a few exceptions such as obs. 3 and obs. 6. In contrast, flow-based TSNPE yields posterior predictives that are less similar to the observations, with a few exceptions such as obs. 1 and obs. 8. These qualitative results further support the quantitative results reported in the main text, demonstrating that TNSPE-PF performs more reliable inference with fewer simulations.

C.4 Pyloric Network Model

A key challenge in this setting is the extreme sparsity of valid simulations. In particular, 99% of parameter samples from the prior yield implausible voltage traces, preventing the computation of summary statistics. For this reason, previous work has relied on millions of simulations for an amortized NPE-based posterior approximation (18 million in Gonçalves et al. [60]). The simulation count could later be reduced to 9 million by restricting the prior to *valid* simulations with an additional classifier [67]. Moreover, several sequential algorithms have been developed to tackle this problem. First, SNVI was proposed, which learns both a likelihood and a variational approximation of the posterior. Notably, the likelihood model was only based on "valid" simulations and corrected by a classifier that predicts whether parameters are valid. This approach made it possible to produce good posterior samples with $3.5 \cdot 10^5$ simulations [18]. This number was reduced even further to only $1.5 \cdot 10^5$ simulations by S-UNLE, which made use of energy-based models and Langevin dynamic MCMC [68]. Later, this approach was extended to SNPE approximations, specifically using TSNPE [37].

Similarly, we want to use only *valid* simulations in the context of NPE-PFN and therefore adaptively truncate the prior using a TabPFN classifier. This approach is based on the intuition that the posterior should place little or no probability mass on regions in parameter space that are likely to lead to invalid simulations [18, 37, 67]. Specifically, as simulations accumulate over rounds, we randomly fill the context of the TabPFN classifier with up to $5 \cdot 10^3$ valid and invalid simulations. The prior is then truncated to $\tilde{p}_{\text{val}}(\boldsymbol{\theta}) \propto \mathbb{I}(P(\text{valid} \mid \boldsymbol{\theta}) > c)p(\boldsymbol{\theta})$, where we choose $c = 0.3$. We then use TSNPE-PFN together with an approximate importance resampling scheme instead of rejection sampling for efficiency. Specifically, we oversample by a factor of $K = 10$ and obtain 10^4 samples, which are then resampled to 10^3 samples based on their importance weight $w(\boldsymbol{\theta}) = \tilde{p}_{\text{val}}(\boldsymbol{\theta})/q_\phi(\boldsymbol{\theta} \mid \boldsymbol{x}_o)$. For evaluation, we exactly follow Glaser et al. [68] by computing the percentage of valid simulations as well as the energy score in each round. Note, however, that the overall experimental setting is not perfectly identical as we use a faster implementation of the pyloric network simulator based on Jaxley [91].

Previous sequential methods started with an initial simulation budget of $5 \cdot 10^4$ prior simulations to acquire a set of ~ 500 valid simulations. Due to the effectiveness of NPE-PFN on small simulation budgets, we start with only $5 \cdot 10^3$ simulations, thus acquiring only ~ 50 valid simulations in the first round. From there, we acquire 10^3 new simulations in each round, updating both NPE-PFN and the restricted classifier with the additional data. We stop after 45 rounds for a total of $5 \cdot 10^4$ simulations. Despite this limited budget, TSNPE-PFN quickly converges to a posterior that produces a high number of valid simulations that closely match the observed measurement. After the final round, we obtain a valid percentage of 96.83% and an energy score of 0.0899 using only $5 \cdot 10^4$ simulations—the starting point of the other methods—and still surpass their final performance (Fig. 5). The evaluation presented in the main text only examines the fidelity of the posterior predictives. Therefore, we include a visualization of the full posterior distribution after the final round (Fig. D-13), which shows broad marginal distributions and similar features as the posteriors reported in previous work [18, 37, 60, 68].

In terms of runtime, most of the computation time is spent on sampling from the restricted prior and running simulations. During the first few rounds, the process is faster (about 2 minutes) due to the smaller context size in TSNPE-PFN. Once the context limit is reached, the time to propose the next 10^3 parameters stabilizes at about 7 minutes, using an Nvidia H100 GPU. Notably, the performance of previous state-of-the-art methods on this task is surpassed after only 10 rounds—an hour of runtime (including simulation time)—which is substantially faster than any previously proposed methods [18, 68].

We choose the default order as the autoregressive order for sampling. The total number of possible permutations in this task is greater than 10^{33}, thus finding the optimal permutation in this space is highly challenging. To gain some insight, we perform an ablation study on 50 distinct random permutations using the context dataset identified by the final-round posterior. We compute the presented metrics with 10^3 simulations. We found that the energy score varies between 0.081 - 0.094 and the valid rates between 96% - 98% (5% and 95% quantile each). These ranges match closely with the results reported in Fig. 5 and demonstrate that there is no significant effect for the autoregressive order in this experiment. Appendix Sec. D.8 reports the same conclusion across all benchmark tasks.

D Additional experiments

D.1 Additional tasks

Here we extend the SBI benchmark experiments (Sec. 3.1) with additional inference tasks from other scientific fields or interesting synthetic tasks used in previous work.

Specifically, the additional tasks include simulators from physics (Weinberg, Stellar Streams), computer science (M/G/1 queue) [16], as well as "structured" synthetic tasks (Tree, HMM) [21]. Out of these, the stellar streams simulator stands out as particularly computationally demanding ($>$ 30 min. per simulation per CPU core; with two-dimensional parameters, and 199 dimensional data). For this reason, we investigate the performance only up to 10^4 simulations.

The tasks in Hermans et al. [16] do not come with a ground truth posterior. We therefore evaluate them only via the average log posterior density (Fig. D-1a). Overall, the results on these additional tasks again demonstrate that, for a small simulation budget, NPE-PFN significantly outperforms NPE. Specifically, NPE-PFN strongly outperforms NPE on tasks with sparse conditional dependencies, i.e., Tree, HMM, Streams, in which it is one to three orders of magnitudes more simulation-efficient than NPE. The stellar streams task is a good example use-case, where this efficiency is particularly beneficial: Attaining 100 simulations is feasible on a consumer-grade CPU (10 cores) in 5 hours, and NPE-PFN achieves a comparable performance to NPE with 10^4 simulations, which would require 20 days. In addition, we investigated the calibration of NPE-PFN for which we observed a similar trend (Fig. D-1b on a subset of tasks). Finally, we performed a direct comparison to reference posteriors where available. We also extended the HMM task to 50 parameter and data dimensions as an additional test for high-dimensional yet "structured" parameter spaces (Fig. D-1c). Here, NPE-PFN generally outperformed NPE.

The HMM example specifically illustrates how NPE (or, more generally, conditional density estimation) fundamentally struggles with dimensionality, independent of the complexity of the simulation process. One approach to address this challenge is to incorporate suitable inductive biases or constraints (such as known conditional dependencies) into the estimation network architecture [21, 30, 92]. The inductive bias from TabPFN as used in NPE-PFN seems to be especially beneficial in such cases. TabPFN was trained exclusively on synthetic datasets generated from random structural causal models (SCMs), which by construction exhibit (sparse) conditional (in-)dependencies. We therefore hypothesize that pretraining on SCMs enables TabPFN to automatically detect and exploit such (in-)dependencies directly from data. A detailed investigation of these mechanisms would be an interesting direction for future work.

D.2 SBI benchmark with more baseline methods

Here, we extend the SBI benchmark experiments (Sec. 3.1) by including two additional baselines: neural ratio estimation (NRE) from Durkan et al. [14] and an ensemble of NPE models, denoted NPE (Ensemble) following Hermans et al. [16]. Ensemble models are known to improve predictive performance [93] and, in the context of SBI, to improve simulation-based calibration (SBC) [16]. We evaluate all methods under the same experimental conditions as before (Sec. 3.1), using five equally weighted ensemble members for NPE (Ensemble).

We present results in terms of C2ST and SBC metrics (Fig. D-2, extending the results from the main text (Fig. 2). For SBC, we use the Error of Diagonal (EoD) to quantify the deviation from perfect calibration. NRE performs similarly or worse on all tasks, while NPE (Ensemble) matches standard NPE in terms of C2ST and achieves the best calibration of all baselines. NPE-PFN is comparable to NPE (Ensemble) in terms of calibration on most tasks, while often outperforming it in

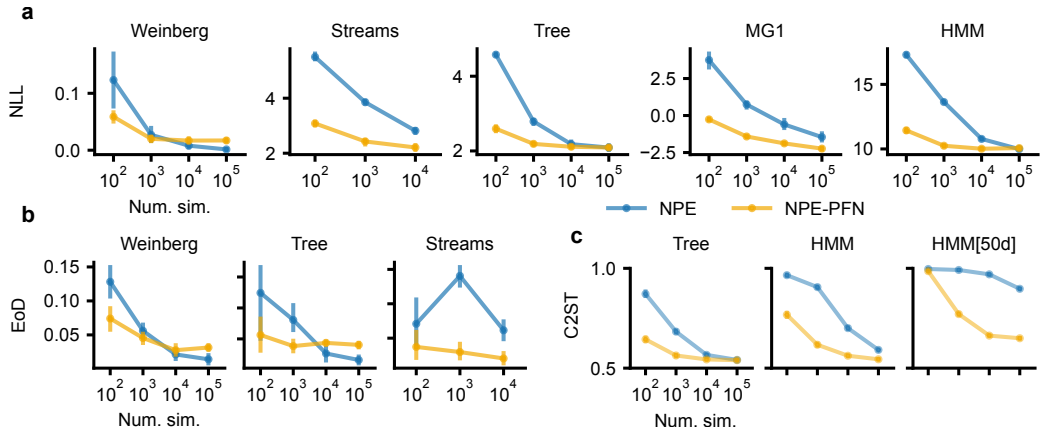


Figure D-1: **Results on additional tasks.** (**a**) Performance under the average negative log likelihood metric (NLL) across five new tasks for NPE and NPE-PFN. (**b**) Calibration error on a subset of the new tasks. (**c**) Performance in C2ST against reference posteriors on tasks for which reference are available. We also investigate a different parameterization of the HMM tasks by increasing dimensionality. Note that the simulation budget of the Streams tasks only goes up to 10^4 simulations.

predictive accuracy and simulation efficiency. Calibration plots (Fig. D-3) corroborate the EoD-based calibration summary with fine-grained information.

These results underscore that NPE-PFN not only achieves strong predictive performance but also provides well-calibrated posteriors that match or exceed ensembles. While we do not directly compare NPE-PFN with the work of Delaunoy et al. [36], who present a SBI method for small simulation budgets using Bayesian neural networks, their method reports modest gains over ensembles, whereas NPE-PFN consistently outperforms them—further emphasizing the effectiveness of our approach with a limited simulation budget.

Throughout this manuscript, the comparisons of NPE-PFN with baseline methods are performed using the sbi library's default hyperparameters [55]. Here, we additionally perform hyperparameter optimization of our main baseline method NPE. For each task and simulation budget, we conduct a random search for flow (flow type, number of flow layers, dimensionality of hidden flow layers) and optimization (batch size, learning rate) hyperparameters. In each setting, we limit the computing time to ten hours. For seven tasks and four simulation budgets, this results in a total computing time of 280 hours.

Compared to NPE with the default settings, NPE (Sweep) achieves better performance in some tasks, particularly with smaller simulation budgets (Fig. D-4). This is due to the shorter training times with 10^2 and 10^3 simulations, enabling the random search to cover large parts of the search space. For 10^4 and 10^5 simulations, no or only minor improvements can be observed. Further improvements for larger budgets would require more computing time and better hyperparameter optimization algorithms, such as Bayesian optimization. Despite the substantial computing time per setting, the optimized NPE (Sweep) still lags behind NPE-PFN in terms of performance for most tasks with smaller budgets (Fig. D-4). Overall, these results again demonstrate NPE-PFN's strong default performance, providing an easy-to-use method that is competitive with, or even superior to, baseline methods where intensive hyperparameter optimization was performed. This strong default performance is particularly beneficial in the sequential setting, where hyperparameter optimization poses significant challenges due to long inference times, changing training conditions across rounds, and the difficulty of selecting suitable metrics for optimization.

D.3 Inference speed

While NPE-PFN is completely training-free, the autoregressive use of TabPFN requires computational effort during inference (Fig. 2b). The inference speed of NPE-PFN depends on three parameters: 1) the number of simulations passed as the in-context dataset; 2) the dimensionality of the observation space (i.e., the number of features); and 3) the dimensionality of the parameter space (i.e., the number of parameters).

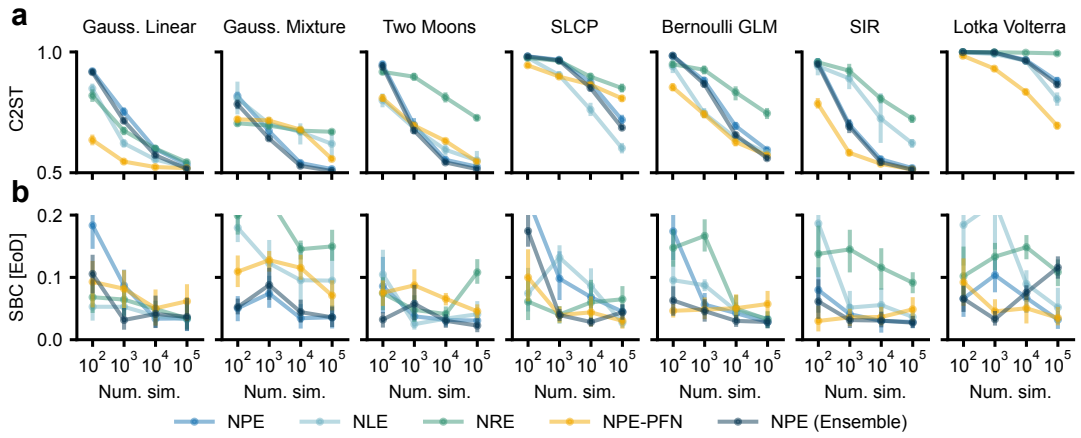


Figure D-2: **Extended SBI benchmark results for amortized NPE-PFN. (a)** Extension of Fig. 2a with more baseline methods, namely NRE and NPE (Ensemble). **(b)** Mean absolute error between the calibration curves and the diagonal (EoD), with 0 indicating perfect calibration. Full calibration curves in Fig. D-3.

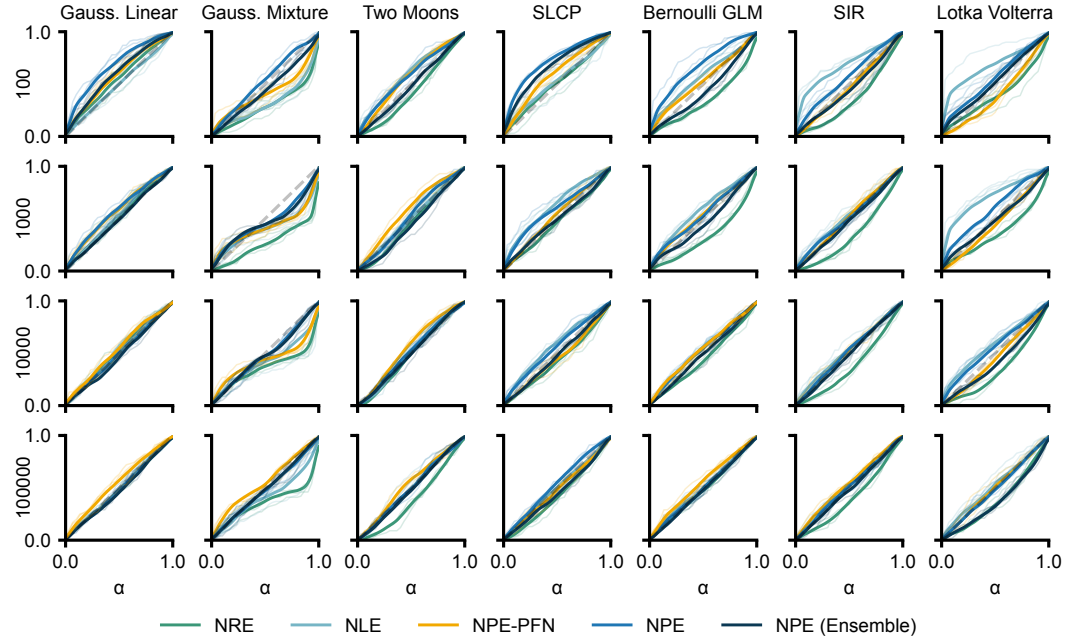


Figure D-3: **Full calibration curves for all tasks.** For each task (columns) and each simulation budget (rows), we plot the associated mean calibration curve (bold) as well as 3 individual runs (transparent, thin).

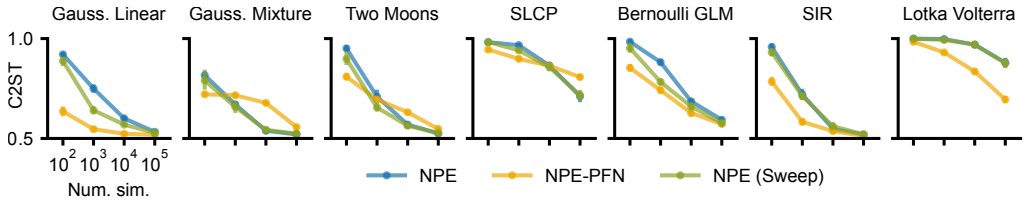


Figure D-4: **SBI benchmark results with NPE hyperparameter optimization.** C2ST for NPE-PFN, NPE with default settings, and NPE (Sweep), where hyperparameter optimization was performed. Specifically flow and optimization hyperparameters are optimized separately for each benchmark task and simulation budget using ten hours of random search.

Here, we provide a thorough examination of the inference speed of NPE-PFN across different simulation budgets and varying dimensionalities of the parameter and observation spaces (Tab. D-2). All results were obtained using an Nvidia A100 GPU. In general, as the value of any of the three parameters increases, inference becomes slower. The dimensionality of the parameter space has the strongest impact on overall inference speed, since the autoregressive use of TabPFN necessitates the reprocessing of the adapted context for each additional parameter dimension. For the largest parameter and observation dimensions and a context of 10^4 simulations, inference can take several minutes. However, this can be alleviated by filtering; for example, by selecting the 10^3 most relevant simulations (see Sec. D.6 for details).

Table D-2: **Inference speed of NPE-PFN.** Inference speed to sample a batch of 10^4 samples from the posterior measured in *seconds* (on an Nvidia A100 GPU). Rows show increasing dimensionality of the parameter space, columns show increasing dimensionality of the observation space, stratified by different simulation budgets.

	10^2 **Sim.**					10^3 **Sim.**					10^4 **Sim.**				
θ \ x	**2**	**4**	**8**	**16**	**32**	**2**	**4**	**8**	**16**	**32**	**2**	**4**	**8**	**16**	**32**
2	6	4	5	6	7	7	6	5	9	8	7	8	10	14	23
4	11	10	10	11	13	10	11	11	13	17	15	19	24	32	48
8	19	23	22	27	30	23	23	24	27	35	35	40	49	66	101
16	45	47	47	53	63	49	52	54	61	77	89	99	116	152	221
32	97	103	107	122	137	115	116	125	141	171	251	272	307	370	516

D.4 Embedding networks for high-dimensional data

We evaluate the performance of **NPE-PFN-Infomax** on high-dimensional simulation-based inference tasks, specifically focusing on a spatial SIR model [16] and an extended Lotka-Volterra task [94]. In the case of the spatial SIR task, where the true posterior is intractable, we rely on indirect metrics such as negative log-likelihood (NLL), TARP [95], and SBC for evaluation. NPE-PFN-Infomax leverages compressed embeddings learned via an infomax objective to distill key information from high-dimensional observations, which are then used as inputs to NPE-PFN.

For the spatial SIR task, we employ a 2D CNN with four convolutional layers consisting of 16, 32, 64, and 128 channels, each with a kernel size of 5 and max pooling of size 2. The resulting feature maps are flattened and passed through a two-layer MLP to produce a 10-dimensional summary statistic. For the extended Lotka-Volterra task (300-dimensional time series), we adopt a similar architecture using a 1D CNN with a kernel size of 3 and an output embedding of 16 dimensions.

Our results (Fig. D-5) indicate that such embedding networks—whether pre-trained or jointly trained—can be used to extend the applicability of NPE-PFN to high-dimensional observations. However, in most cases, this approach does not outperform an end-to-end trained NPE baseline. Interestingly, TabPFN without an embedding network (when applicable) often achieves better performance, suggesting that the compression step, independent of NPE-PFN, may be suboptimal. One possible explanation is that TabPFN, having been trained on causally structured data, can exploit underlying dependencies to improve inference accuracy. In contrast, compressing such highly structured data into low-dimensional summaries may discard critical information, limiting the effectiveness of inference.

D.5 Comparison of autoregressive and ratio-based density evaluation

In Sec. 2.4, we present a method for fast density evaluation based on density ratios, requiring only a single forward pass through the TabPFN classifier. While this approach offers a substantial computational advantage, it is inherently approximate, raising the question of how it compares to the autoregressive density evaluation. To answer this question, we compare the two methods on two tasks—the 2D Two Moons tasks and the 10D Gaussian Linear task—across varying simulation budgets. In each case, we evaluate the density of posterior samples from the trained model given an observation, using 5 random seeds. To construct the dataset for the ratio estimator, we draw $5 \cdot 10^3$ samples each from the posterior of interest and the uniform base distribution, which is the default setting.

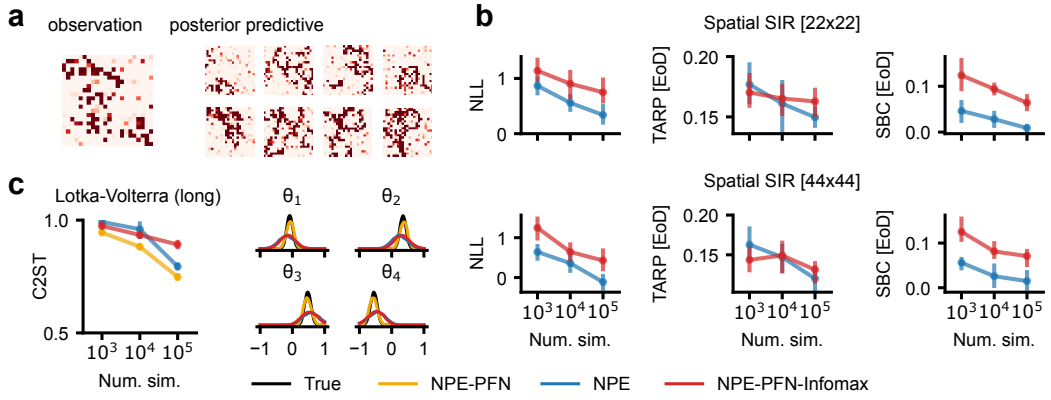


Figure D-5: **High-dimensional data with embedding nets.** **(a)** Visualization of observations in the spatial SIR task alongside posterior predictive samples obtained using NPE-PFN-Infomax. **(b)** Performance evaluation across high-dimensional variants of the spatial SIR task over 5 independent runs, reported in terms of the negative log-likelihood (NLL) of the true parameter and the area off the diagonal in TARP and SBC calibration analyses. **(c)** An instance of the Lotka-Volterra task with an extended time series (300 dimensions), evaluated using NPE-PFN and NPE-PFN-Infomax.

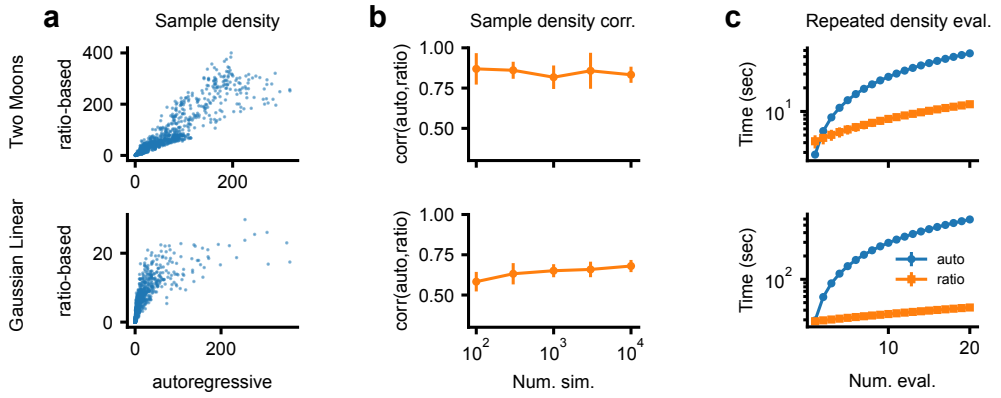


Figure D-6: **Comparison of density evaluation approaches on Two Moons and Gaussian Linear task.** **(a)** Correlation between density estimates from the autoregressive and ratio-based approaches for two observations using 10^4 simulations. **(b)** Pearson correlation between the two density estimates across varying simulation budgets for both tasks. **(c)** Evaluation time (log scale) for an increasing number of density evaluations using an Nvidia H100 GPU, showing the crossover point where the ratio-based approach becomes more efficient.

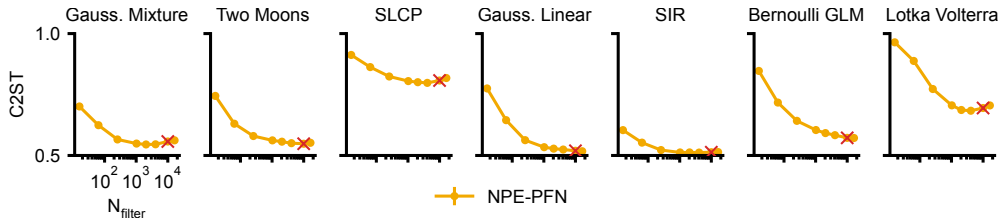


Figure D-7: **SBI benchmark results with varying number of filtered simulations.** C2ST performance of filtered NPE-PFN as a function of the number of filtered simulations N_{filter}. For all tasks, a simulation budget of 10^5 is used. The red cross marks the default choice of $N_{\text{filter}} = 10^4$.

We observe a strong correlation between autoregressive and ratio-based density evaluations for the Two Moons task and a somewhat weaker correlation for the higher-dimensional Gaussian Linear task (Fig. D-6a for a simulation budget of 10^4). This trend persists across simulation budgets (Fig. D-6b). In terms of runtime, the ratio-based method has an initial cost due to posterior sampling, making it slower for the first evaluation. However, once samples are obtained, subsequent evaluations are significantly faster—an advantage that becomes very noticeable as the number of density evaluations increases (Fig. D-6c; note the logarithmic y-axis).

Therefore, in scenarios that require repeated density evaluations for a single observation—such as rejection sampling in TSNPE-PFN—the ratio-based approach offers a highly favorable trade-off: It provides sufficient accuracy at a fraction of the computational cost.

D.6 Varying the number of filtered simulations

When filtering simulations, we always make full use of TabPFN's recommended maximal context size of 10^5 data points. However, in principle, the number of simulations N_{filter} selected by the filter is a hyperparameter that can be optimized. Here, we investigate the impact of this hyperparameter by evaluating the performance of NPE-PFN on the SBI benchmark tasks and a simulation budget of 10^5. Specifically, we vary the number of filtered simulations N_{filter}, setting it to 16, 64, 256, 1024, 2048, 4096, and 16384, and compare the resulting performance to our default choice of 10^4.

For small values of N_{filter}, the performance of NPE-PFN deteriorates across all tasks (Fig. D-7). In contrast, for $N_{\text{filter}} = 1024$ or larger, performance is very similar to that of our default choice with 10^5 simulations. For some tasks, a slight decrease in performance is observed for the largest value of N_{filter}, likely because this exceeds the recommended context size of TabPFN. Importantly, our default choice is always optimal or near-optimal. These results suggest that the recommended maximal context should be utilized fully. Nevertheless, these results also indicate that good performance can be achieved with smaller filter sizes (e.g., $N_{\text{filter}} = 2048$) to reduce the computational load.

D.7 Varying feature and noise distributions

Here, we investigate the robustness of NPE-PFN to different types of noise. To evaluate the robustness in such cases, we construct a variant of the Gaussian linear task from Lueckmann et al. [27] with non-Gaussian features and noise

$$\boldsymbol{\theta} \sim p_{\text{feat}}(\boldsymbol{\theta}), \tag{D-8}$$

$$\boldsymbol{x} \sim \theta + p_{\text{noise}}(\boldsymbol{x} \mid \boldsymbol{\theta}). \tag{D-9}$$

Both $p_{\text{feat}}(\boldsymbol{\theta})$ and $p_{\text{noise}}(\boldsymbol{x} \mid \boldsymbol{\theta})$ are chosen from a set of distributions with varying support and tail behavior:

- Cauchy distribution $\text{Cauchy}(0, s)$,
- Laplace distribution $\text{Laplace}(0, s)$,
- Logitnormal distribution $\sigma(\mathcal{N}(0, s^2))$ with $\sigma(x) = \frac{1}{1+e^{-x}}$,
- Normal distribution $\mathcal{N}(0, s^2)$,
- Student's t-distribution $t_5 \cdot s$ where t_5 represents a Student's t-distribution with 5 degrees of freedom,

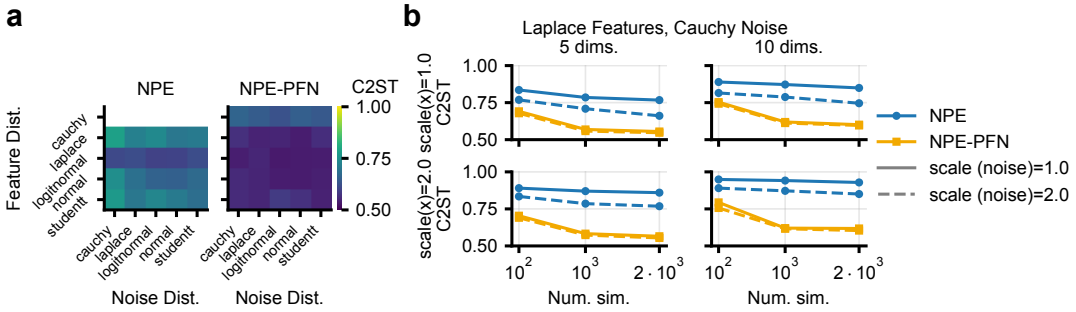


Figure D-8: **Robustness to feature and noise distributions. (a)** Heatmaps showing C2ST performance for all combinations of feature and noise distributions with $\dim(\boldsymbol{\theta}) = 5$, $s = 1.0$, and 10^3 training simulations. White cells indicate failed training. **(b)** C2ST across training set sizes for an example combination of distributions. These panels are zoom-ins from (a), highlighting the stability of NPE-PFN under varying distributional assumptions.

where s is the scale parameter. We vary the feature dimension, the scale of feature and noise distributions, and the number of training simulations, and compare the performance of NPE-PFN against NPE.

We first evaluate performance via C2ST across all combinations of feature and noise distributions for $s_{\text{features}} = s_{\text{noise}} = 1.0$, $\dim(\boldsymbol{\theta}) = 5$, and 10^3 simulations. The heatmap shows that NPE-PFN outperforms NPE consistently, with while cells indicating training failure primarily for NPE (Fig. D-8a). For the example of Laplace feature and Cauchy noise distributions, we provide results as a function of simulation budget, confirming the robustness of NPE-PFN across distributional shifts (Fig. D-8b). Notably, while NPE achieves performance drops on several non-Gaussian configurations, NPE-PFN maintains stable performance.

These results suggest that, although TabPFN is pre-trained using priors based on structural causal models (SCMs) with uniform or Gaussian root noise, nonlinear transformations in the SCM induce rich marginal distributions and allow generalization to a wide range of distributions. NPE-PFN inherits this robustness, and its performance is largely invariant to different feature or noise distributions. Failures of NPE, especially with Cauchy features, are caused by instability in z-scoring, which NPE-PFN avoids through default preprocessing such as the Yeo–Johnson transform [96]. Thus, standard NPE performance could probably be improved by applying appropriate transformations. Because NPE-PFN takes advantage of the automatic preprocessing performed within TabPFN, it provides reliable performance without manual preprocessing, making it a robust and user-friendly solution in practical settings.

D.8 Order of autoregressive sampling

To sample from multi-dimensional (conditional) distributions with TabPFN, we use it in an autoregressive manner. An important question is whether the order in which we sample the dimensions matters for, e.g., the quality of the inferred posterior distributions. To investigate this question, we rerun NPE-PFN on the benchmark tasks from Sec. 3.1, but permute the order in which we sample the parameter dimensions. Two of the seven benchmark tasks are permutation invariant by design (Gaussian Mixture and Gaussian Linear), and we replace these with tasks where a distribution is constructed autoregressively. The first is a simple nonlinear task given by

$$[x_1, x_2] \sim \mathcal{N}(0, 1), \tag{D-10}$$

$$y_1 \sim \mathcal{N}(x_1, 1), \tag{D-11}$$

$$y_2 \mid y_1 \sim \mathcal{N}(\sin(y_1 + x_2), 1), \tag{D-12}$$

$$y_3 \mid y_1, y_2 \sim \mathcal{N}(y_2^2 + y_1, 1), \tag{D-13}$$

$$y_4 \mid y_1, y_2, y_3 \sim \mathcal{N}(y_1 y_2 + y_3, 1). \tag{D-14}$$

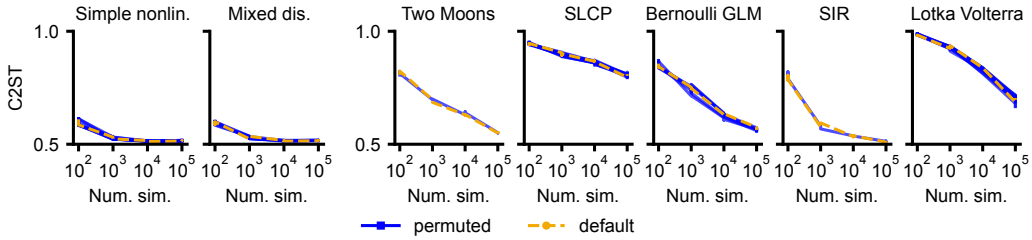


Figure D-9: **Autoregressive ordering.** C2ST across varying simulation budgets for two additional synthetic tasks, and five tasks from the SBI benchmark that are not permutation invariant. Blue lines indicate the average C2ST accuracy over three seeds for up to ten (random) permutations of the sampling order. Dashed orange lines indicate the default order.

The second is a mixed distribution task, given by

$$x = [x_1, x_2] \sim U(-2, 2), \tag{D-15}$$

$$y_1 \sim \text{Gamma}(\text{shape} = 1 + |x_1|, \text{scale} = 1), \tag{D-16}$$

$$y_2 \mid y_1 \sim \text{Uniform}(0, y_1 \cdot 2 + |x_2|), \tag{D-17}$$

$$y_3 \mid y_1, y_2 \sim \text{Beta}(\alpha = 1 + y_1, \beta = 2 + y_2). \tag{D-18}$$

As in Sec. 3.1, we compute the C2ST with respect to the ground truth for evaluation. We run NPE-PFN for the default autoregressive order (as defined in the SBI benchmark or in the equations above) and up to ten (random) permutations over three random seeds. Note that tasks with two or three dimensions have only two or six possible permutations, respectively (including the default one), so random subsampling is not required.

Across all benchmark tasks and simulation budgets, NPE-PFN achieves nearly identical performance across the default and permuted orders (Fig. D-9). That is, for these benchmark tasks, the performance of NPE-PFN is not affected by permuting dimensions. These results show that NPE-PFN is not sensitive to the order in which the parameters are sampled. As a result, users need not worry about providing NPE-PFN with an "optimal" order. While differences in performance may be possible for very high dimensions or artificial examples, for the applications we consider here, an arbitrary permutation is sufficient.

D.9 Unconditional density estimation on the UCI datasets

Here, we apply TabPFN on some classical *unconditional* density estimation benchmark tasks from the UCI repository [56] as used in several other works [41, 42, 97]. Note that while we still perform density estimation here, it does not involve posterior distributions. Thus, we do not refer to this approach as NPE-PFN. Specifically, we consider the Gas, Power, Hepmass, and Miniboone datasets. These tabular datasets range in dimensionality from 6 to 43 features and contain between $3.1 \cdot 10^4$ and over 1 million samples. We here investigate the unconditional density estimation performance of TabPFN in the low sample regime in comparison to a neural spline flow (NSF) [42]. As in previous works, we compute the negative log-likelihood (NLL) under a held-out test set. While we evaluate on the full test sets, we only use 10^3, 10^4, or (if applicable) 10^5 samples for training. For the two larger settings, we use partitioning (Sec. B.4) with 10 clusters when estimating densities with TabPFN.

For 10^2 and 10^3 samples, TabPFN achieves a smaller negative log-likelihood on all four datasets (Fig. D-10). Similarly, for the 10^4 samples, TabPFN outperforms NSF on all but the power dataset. Interestingly, for the lower dimensional datasets, clustering further reduces the NLL despite having access to the same total number of samples. On the other hand, for the higher dimensional datasets, clustering increases the NLL. For 10^5 samples (or $3.1 \cdot 10^4$ samples for the Miniboone dataset), clustering improves the NLL over any 10^4 sample setting because it allows TabPFN to access a larger number of samples. However, NSF performs better than or equal to TabPFN on all datasets except the Gas dataset at 10^5 samples.

These results suggest that TabPFN is a capable (unconditional) density estimator, especially in the low sample regime.

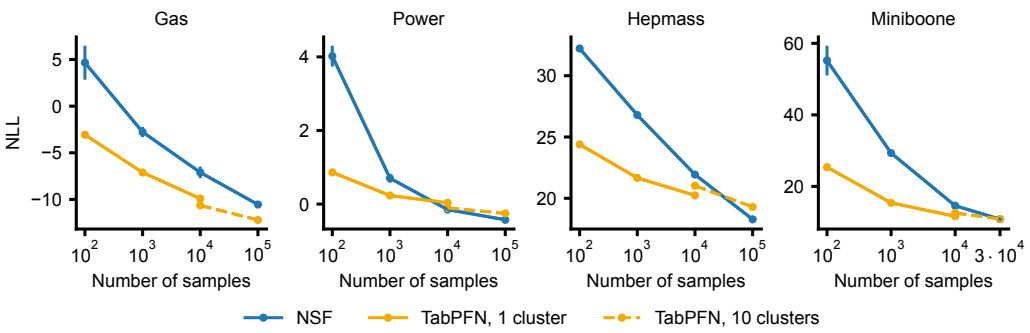


Figure D-10: **Results for unconditional density estimation on the UCI datasets.** Negative-log-likelihood (NLL) for TabPFN (with 1 and 10 clusters) and the neural spline flow (NSF) across the different UCI datasets. Dots indicate averages, and bars show standard deviation over five independent runs.

Figure D-11: **Posterior predictives of TSNPE-PFN** for real observations from the Allen cell type database and a simulation budget of 10^4. Note that the inference is not performed directly on the action potential time series but on seven summary statistics computed from it.

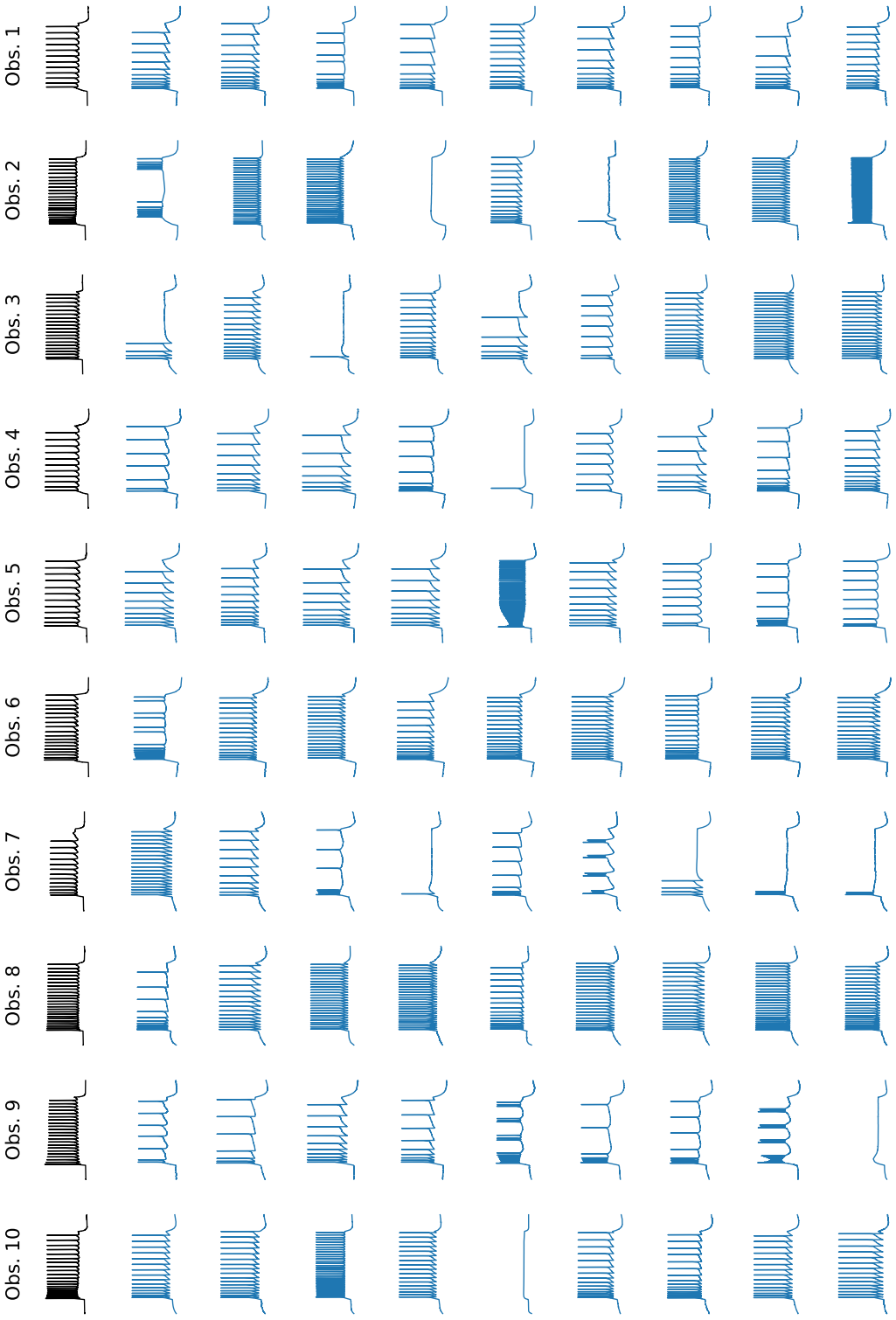


Figure D-12: **Posterior predictives of the TSNPE baseline** for real observations from the Allen cell type database and a simulation budget of 10^4. Note that the inference is not performed directly on the action potential time series but on seven summary statistics computed from it.

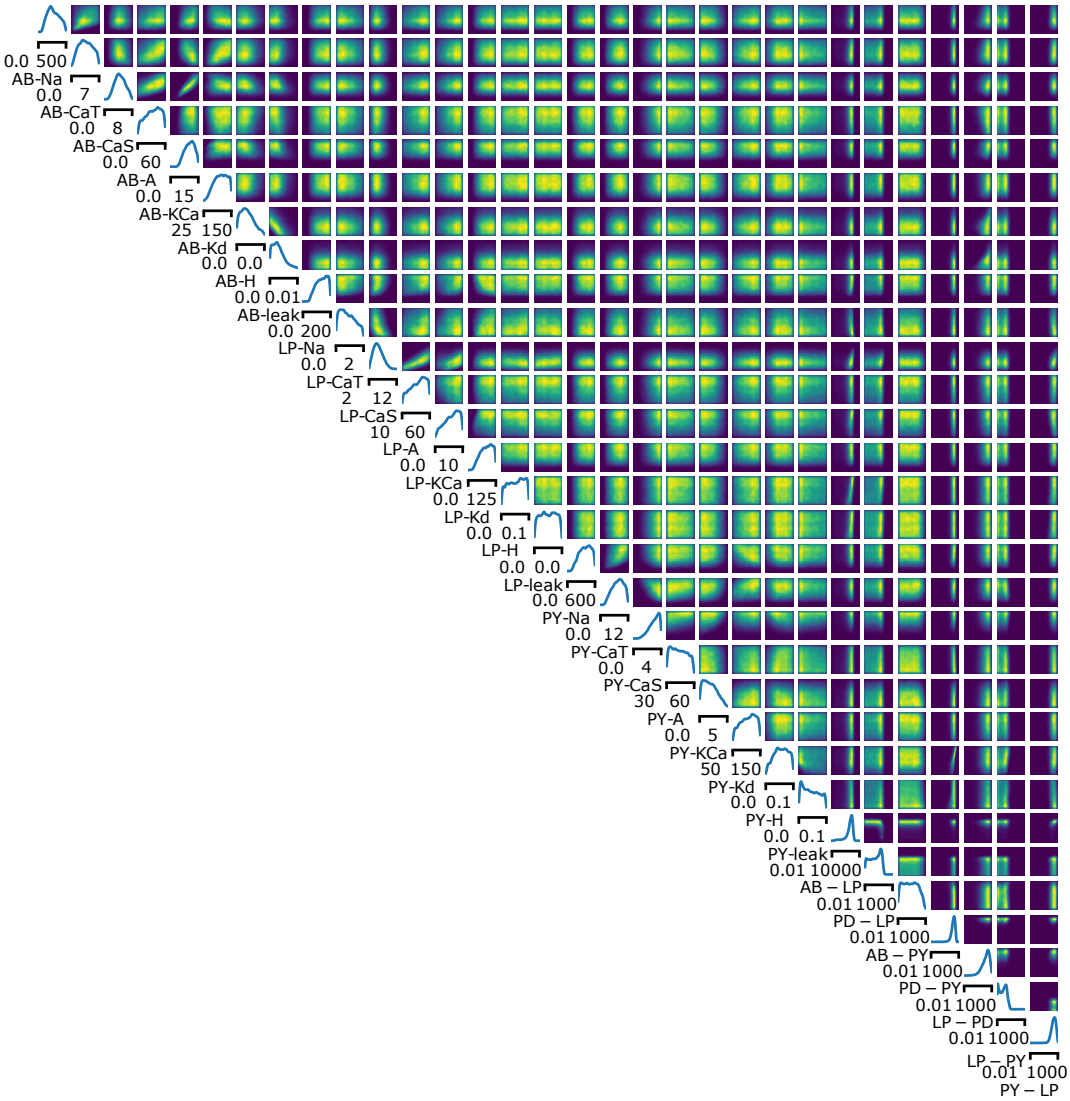


Figure D-13: **Pyloric simulator.** Posterior distributions for all 31 parameters estimated by TSNPE-PFN.