GANs schön kompliziert: Applications of Generative Adversarial Networks

DISSERTATION

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

vorgelegt von
POORNIMA RAMESH
aus Chennai, Indien

Tübingen
2022

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

Tag der mündlichen Qualifikation: 09.12.2022

Dekan:	Prof. Dr. Thilo Stehle
1. Berichterstatter:	Prof. Dr. Jakob H. Macke
2. Berichterstatter:	Prof. Dr. Fabian Sinz

Abstract

Scientific research progresses via model-building. Researchers attempt to build realistic models of real-world phenomena, ranging from bacterial growth to galactic motion, and study these models as a means of understanding these phenomena. However, making these models as realistic as possible often involves fitting them to experimentally measured data.

Recent advances in experimental methods have allowed for the collection of large-scale datasets. Simultaneously, advancements in computational capacity have allowed for more complex model-building. The confluence of these two factors accounts for the rise of machine learning methods as powerful tools, both for building models and fitting these models to large scale datasets.

In this thesis, we use a particular machine learning technique: generative adversarial networks (GANs). GANs are a flexible and powerful tool, capable of fitting a wide variety of models. We explore the properties of GANs that underpin this flexibility, and show how we can capitalize on them in different scientific applications, beyond the image- and text-generating applications they are well-known for.

Here we present three different applications of GANs. First, we show how GANs can be used as generative models of neural spike trains, and how they are capable of capturing more features of these spike trains compared to other approaches. We also show how this could enable insight into how information about stimuli are encoded in the spike trains. Second, we demonstrate how GANs can be used as density estimators for extending simulation-based Bayesian inference to high-dimensional parameter spaces. In this form, we also show how GANs bridge Bayesian inference methods and variational inference with autoencoders and use them to fit complex climate models to data. Finally, we use GANs to infer synaptic plasticity rules for biological rate networks directly from data. We then show how GANs be used to test the robustness of the inferred rules to differences in data and network initialisation.

Overall, we repurpose GANs in new ways for a variety of scientific domains, and show that they confer specific advantages over the state-of-the-art methods in each of these domains.

Zusammenfassung

Die wissenschaftliche Forschung schreitet durch Reproduktion voran. Forscher versuchen, realistische Modelle von realen Phänomenen zu erstellen, vom Bakterienwachstum bis zur galaktischen Bewegung, und untersuchen diese Modelle, um diese Phänomene zu verstehen. Um diese Modelle so realistisch wie möglich zu gestalten, müssen diese jedoch oft an experimentell gemessene Daten angepasst werden.

Jüngste Fortschritte bei den experimentellen Methoden haben es ermöglicht, große Datensätze zu sammeln. Gleichzeitig haben Fortschritte bei den Rechenkapazität eine komplexere Modellbildung ermöglicht. Das Zusammentreffen dieser beiden Faktoren hat dazu geführt, dass sich Methoden des maschinellen Lernens zu leistungsfähigen Werkzeugen entwickelt haben, sowohl für die Erstellung von Modellen als auch für die Anpassung dieser Modelle an große Datensätze.

In dieser Arbeit verwenden wir eine besondere Technik des maschinellen Lernens: generative adversarische Netzwerke (GANs). GANs sind ein flexibles und leistungsstarkes Werkzeug, mit dem sich eine Vielzahl von Modellen an Daten anpassen lässt. Wir untersuchen die Eigenschaften von GANs, die diese Flexibilität untermauern, und zeigen, wie wir sie in verschiedenen wissenschaftlichen Anwendungen nutzen können, die über die bekannten Bild- und Textgenerierungsanwendungen hinausgehen.

Hier stellen wir drei verschiedene Anwendungen von GANs vor. Zunächst zeigen wir, wie GANs als generative Modelle neuronaler Spike Trains verwendet werden können und wie sie im Vergleich zu anderen Ansätzen in der Lage sind, viel mehr Merkmale dieser Spike Trains zu erfassen. Wir zeigen auch, wie dies einen Einblick in die Art und Weise ermöglichen könnte, wie Informationen über Stimuli in den Spike Trains kodiert werden. Zweitens zeigen wir, wie GANs als Dichteschätzer verwendet werden können, um simulationsbasierte Bayes'sche Schlussfolgerungen auf hochdimensionale Parameterräume auszuweiten. In dieser Form zeigen wir auch, dass GANs eine Brücke zwischen Bayes'schen Inferenzmethoden und Variationsinferenz mit Autoencodern schlagen und sie zur Anpassung komplexer Klimamodelle an Daten verwenden. Schließlich verwenden wir GANs, um synaptische Plastizitätsregeln für biologische Geschwindigkeitsnetzwerke direkt aus Daten abzuleiten. Anschließend zeigen wir, wie GANs verwendet werden können, um die Robustheit der abgeleiteten Regeln gegenüber Unterschieden in den Daten und der Initialisierung des Netzwerks zu testen.

Insgesamt setzen wir GANs auf neue Art und Weise für eine Vielzahl von wissenschaftlichen Bereichen ein und zeigen, dass sie in jedem dieser Bereiche spezifische Vorteile gegenüber den modernsten Methoden bieten.

Acknowledgements

A PhD can be a solitary endeavour, but it took an army to get through mine. I owe the completion of it and a world of gratitude to the following people:

First and foremost, to Prof. Dr. Jakob Macke who has been unwavering in his support, encouragement and guidance both scientifically and otherwise – I would have torn up every single piece of work and deleted every experiment I logged on WandB, if not for him constantly reminding me that failure and scientific discovery are intertwined, and one is worth the other.

To Giacomo Bassetto, who showed me what doing scientific research meant, and how to not to compromise on those principles (even though I was not the easiest grasshopper to his Master Po).

To Dr. Marcel Nonnemmacher, Artur Speiser and Dr. Jan-Matthis Lückmann who taught me the nuts and bolts of scientific research.

To Dr. Pedro Gonçalves: official and unofficial mentor, agony aunt and an extraordinary human being.

To Auguste Schulz, Tamara Müller and Harsha Dixit, who are always inspiring and sometimes, the only light in the darkness.

To Jan Boelts (with his fantastic memes and puns, including the title of this thesis), Michael Deistler, Dr. Richard Gao, Janne Lappalainen, Avleen Sahni and Florian Müller, who turned all potentially nightmarish experiences into some of my most cherished memories.

To Julius Vetter, Basile Confavreux, Jan Boelts, Auguste Schulz, Janne Lappalainen, Dr. Richard Gao, Dr. Pedro Gonçalves, for the whacky scientific discussions, and feedback on the thesis.

To Franziska Weiler and Alexandra Petelski – the rest of us are able to focus on the science only because they take care of the adulting.

And finally, to my parents, sister and grandmother who have been understanding of every mad choice, and have cheered me on every step of the way.

Contents

Acronyms

ABC Approximate Bayesian Computation.

ELBO Evidence Lower Bound Optimization.

GANs Generative Adversarial Networks.

GATSBI Generative Adversarial Training for Simulation-Based Inference.

JSD Jensen-Shannon Divergence.

MCMC Markov Chain Monte Carlo.

MLE Maximum Likelihood Estimation.

SBI Simulation-Based Inference.

VAEs Variational Autoencoders.

Chapter 1

Introduction

Douglas Adams in the *Hitchhikers' Guide to the Galaxy* said, "If you try and take a cat apart to see how it works, the first thing you have on your hands is a non-working cat." Despite that accurate observation, scientists routinely try to take apart real-world systems to see how they work. For example, scientists disrupt DNA transcription [78], interfere with insect and animal social hierarchies [45, 161], build massive interferometers [62] and much more, to try and understand a variety of natural phenomena. Perturbing the normal functioning of a system in the hopes of trying to gather data about its behaviour is the established scientific method. However, gathering data about these systems requires observing them under perturbation, and measuring various quantities relevant to the perturbation we want to study. Sophisticated observation methods abound from microscopy[138] to observe biological and physical processes at atomic and molecular length-scales to long-term tracking devices for monitoring the behaviour of groups of animals [14], extra-solar planet detection [183] and plate-tectonics [48].

However, observing systems under perturbation is not the only avenue of enquiry into real-world systems: an alternative is model building. Constructing simplified mathematical models to explain observed phenomena has long been a mainstay of science. These models are typically set up to produce behaviour that resembles a real-world system, and contain tunable parameters that effectively allow us to control how the model behaves. For example, a Newtonian model of planets revolving around the sun generates orbital trajectories, and has parameters that include the gravitational constant and the masses of the planet and the sun. The parameters of the model, their relationship to each other and the outputs (or behaviour) of the model together constitute a hypothesis about the mechanisms underlying real-world phenomena. Recent advances in computational power have allowed us to push the boundaries of modelling techniques beyond the regime where only tractable and analytical models could be studied. Almost all scientific fields these days rely on realistic models of natural phenomena as a means of understanding these phenomena.

While model building and observational methods independently provide plenty of scientific insight, they are most useful when they each inform the other. To this end, we attempt to "fit" our models to observed experimental data i.e. we use our experimental measurements to edify and improve the models we build. In order to do this effectively, we construct models that can mimic the experimental conditions under which we perturb the real-world system i.e. the model is explicitly set up to include the aspects of the system that we perturb, and also to produce the same data we observe. For example, the Hodgkin-Huxley model of the neuron generates cell membrane voltages, which we can measure experimentally [66]. In the model, the cell membrane is also influenced by input currents, which Hodgkin and Huxley used to experimentally elicit membrane voltage changes in squid neurons. Models that replicate experimental conditions are effectively *testable*

hypotheses about natural phenomena: we can now compare the data that we get from perturbing the model, to the data that we collect from perturbing the real system. Any similarity between the two sets of data is evidence for the hypothesis embodied by the model. Furthermore, we can tune the parameters of the model until it produces output that is identical to observed data under some measure of similarity (or objective function) – this is model fitting. For example, Hodgkin and Huxley fixed the parameters of their model (ion channel conductances and membrane time constants) such that their model produced the exact changes in voltage traces with input currents that they obtained experimentally. Finally, a fitted model can be further interrogated, either in ways that natural systems cannot be, or to propose further experiments or hypothesis to gain scientific insight – Hodgkin and Huxley hypothesized the existence of an ion channel with 4 protein subunits from their model.

The natural world however, is influenced by factors that we may be unable to pinpoint, perturb or measure e.g. it is hard to accurately predict how interacting prey-predator populations evolve without taking into account the complex interactions between the prey or predator and other animals and plants that make up their environment. More precisely, these factors cause our experimental measurements to vary even when we do not perturb the system, or vary disproportionately to a perturbation. This means that there is a need for more and more sophisticated models that capture, in addition to changes in the system under perturbation, the variability in these phenomena. However, the endeavour to capture these additional factors of variability in the model also lead to them being more complex, both to set up and to study. It also leads to greater difficulty in fitting these models to data. Fitting these models to data is non-trivial, both due to the dimensionality of the model parameters and the outputs as the models grow in complexity, as well as the difficulty in defining an objective function to tune the model parameters (i.e. train the model) to capture all of these desired features of variability.

In this thesis, we show how machine learning methods, and specifically, generative adversarial networks [50, GANs] are effective in tackling the challenges presented by these models. GANs are renowned for being able to generate realistic data – the goal of most model-building endeavours. Furthermore, GANs are well-suited for generating realistic, high-dimensional data such as images or video, and are capable of of encapsulating complex data-generating mechanisms. Finally, they can be fit directly to observed data, without the need for a well-defined objective function. This makes them applicable to a wide-range problems. We demonstrate how these properties can be exploited when dealing with the challenge of fitting sophisticated, high-dimensional, stochastic models to observed data.

We present some applications where complex and stochastic scientific models need to be set up and fit to observed data. The first application concerns accurately reproducing neural population spike trains, by capturing both signal-driven responses in neurons, as well as latent noise sources that lead to variability in the responses. The second application deals with capturing how variability in the parameter settings of a scientific model contributes to variability in its outputs, using Bayesian inference. The third application involves inferring the mechanisms of weight changes in plastic networks of neurons from observations of neural activity. All three applications deal with complex models attempting to capture real-world phenomena, that incorporate stochasticity in order to capture variability in the model outputs. All of these models are further complicated by the dimensionality of the model parameters and outputs (e.g. climate models are high-dimensional because of the spatial and temporal extent of the model parameters and inputs), the nature of the model output (e.g neural population responses are binary-valued, which makes defining an objective function based on variability in the responses difficult) and the necessity of defining an objective function to fit these models (e.g. defining an objective over observed data that captures

information about the underlying plasticity mechanism for the neural population). We show how the flexibility of the GAN approach can be extended to tackle these challenges.

In the following chapters, we first introduce some relevant machine learning concepts (section 2.1), and then present an overview of GANs (section 2.2). We then present our algorithm using GANs for reproducing neural data (chapter 3), for Bayesian inference (chapter 4) and for inferring plasticity rules from data (chapter 5). With these applications, we demonstrate how GANs enable and extend our model-fitting capabilities, and how their use engenders scientific insight into the domains outlined above.

Chapter 2

General Background

In this chapter, we provide an overview of some of the concepts that are relevant for the work described in this thesis.

2.1 Overview of probabilistic machine learning concepts

Deep learning can be thought of as a set of tools for learning complex and flexible input-output functions from data i.e. we learn a mapping f from inputs x to targets y: $y = f(x)$. These functions are implicitly represented by neural networks with parameters ϕ: $y = f_\phi(x)$, that are learnt by optimizing a performance measure for this function f i.e, an objective function or error function. Some examples include mapping temperature to disease prevalence [28], images to classes [71, 88, 160], stimuli to neural activity [11, 110, 141]: machine learning is widely applicable precisely due to this potentially infinite range of mappings that neural networks are capable of representing.

When all quantities of interest (x, y) are considered random variables: i.e. the outcome of random events, learning mappings between x and y is probabilistic machine learning. In this case, the neural networks are statistical models of a stochastic process that predict y from x. While deterministic models are useful when we wish to reproduce y from x in precise detail, in most real-world applications we interested in capturing some global features of the relationship between x and y, rather than strictly reproducing y from x. Moreover, real-world observations of x and y, which we use to train our models, are always noisy. Thus, we would also need to represent the uncertainty in the mapping $y = f_\phi(x)$ due to this noise. In these cases, probabilistic machine learning is advantageous. Generative adversarial networks[50, Generative Adversarial Networks (GANs)], the workhorse of this thesis, are an example of a probabilistic machine learning approach.

2.1.1 Generative models

In this thesis, we deal almost exclusively with generative models of data. We use "generative" in the sense that the models we deal with produce some target data y given labels or inputs x, rather than inferring labels x given data y. While this distinction is rather blurry and depends strongly on our definition of label and target, it is nevertheless useful, since the variables we are interested in for the problems considered in this thesis are generated by the model, rather than input to them.

2.1.2 Supervised models

In supervised learning, we can define an objective function $\mathcal{L}$ that measures the discrepancy between the outputs of the function or neural network $\hat{y}$ and the groundtruth targets y i.e. $\mathcal{L}(\hat{y}, y)$, in closed form. This error function is a notion of "distance" between the outputs of the network $\hat{y}$ and the targets y that we want it to produce. We use this objective function to update the parameters of the network, such that the $\mathcal{L}(\hat{y}, y)$ is minimized i.e., we compute the gradient of the objective function with respect to the parameters $\nabla_\phi \mathcal{L}(\hat{y}, y)$, and use it to update the parameters ϕ:

$$\begin{aligned} \hat{y} &= f_\phi(x) \\ \phi &\longleftarrow \phi - \nabla_\phi \mathcal{L}(\hat{y}, y) \\ \phi^* &= \min_\phi \mathcal{L}(\hat{y}, y) \end{aligned}$$

Classification, regression and risk minimization [20] are all examples of supervised learning problems.

Maximum likelihood estimation

Since in probabilistic machine learning, x and y are random variables, we can define a likelihood distribution $p(y|x)$ i.e. the conditional distribution of targets y, given inputs x. If we think of the mapping $y = f(x)$ as representing the likelihood, then the neural network mapping $\hat{y} = f_\phi(x)$ represents an approximation to the likelihood $q_\phi(y|x)$. Maximum Likelihood Estimation (MLE) allows us to learn the parameters ϕ, such that the ground-truth data y has the highest probability given in put x, under $q_\phi(y|x)$. Thus, MLE is a special case of supervised learning, where the loss function is the negative log likelihood of the outputs y given the inputs $q_\phi(y|x)$ under the model defined by the neural network f_ϕ. In other words:

$$\begin{aligned} \hat{y} &\sim p(\hat{y}|f_\phi(x)) = q_\phi(\hat{y}|x) \\ \phi^* &= \min_\phi \; -\log q_\phi(y|x) \\ &= \max_\phi \; \log q_\phi(y|x) \end{aligned} \tag{2.1}$$

Linear regression can be recast as an MLE problem, in which the network outputs are Gaussian distributed i.e $y \sim \mathcal{N}(f_\phi(x), 1)$, and logistic regression is a form of MLE in which the network outputs are Bernoulli random variables ($y \sim \text{Bernoulli}(f_\phi(x))$.

Latent variable models

Latent variable models incorporate latent variables into the statistical model approximating the likelihood of the targets given the inputs, in order to capture correlations in the data that are not driven by the inputs. This nescessitates incorporating additional random variables $r \sim p(r)$ into the model, and integrating over them to obtain the likelihood:

$$\begin{aligned} & \hat{y} = g_\phi(x, z, r), \; r \sim p(r) \\ \implies & \hat{y} \sim q_\phi(\hat{y}|x, r) \\ q_\phi(\hat{y}|x) &= \int q_\phi(\hat{y}|x, r) \; p(r) \; dr \end{aligned} \tag{2.2}$$

Computing this integral is non-trivial, since the product of the distributions $q_\phi(\hat{y}|x,r)$ and $p(r)$ can be computed in closed-form, only when the two are conjugate distributions [124].

Factor analysis [61] circumvents computing this integral by assuming that the variability due to these latents indicates structure in a manifold that has fewer dimensions than the data. These methods attempt to capture this manifold using dimensionality reduction methods such as principal component analysis [137, PCA] and independent component analysis [73, ICA].

Latent dynamical systems analysis[105, 130, 193] also learns the low-dimensional manifold from data and relies on MLE for learning the model parameters. These methods assume that the reduced-dimension latents vary over time, and capture the dynamics using either Gaussian Processes [149, GPs] or flow equations:

$$\hat{y}_t = g_\phi(x_t, z_t) + Cr_t \tag{2.3}$$
$$r_t = f_\theta(r_{t-1}) + \epsilon, \epsilon \sim p(\epsilon) \tag{2.4}$$

Note that these models typically assume Markov dynamics [46]: i.e. the activity y_t is directly influenced only by r_t and not by latents from previous timesteps. This makes fitting the parameters ϕ, θ and C using MLE – which requires computing the integral in Equation 2.2 – easier. These models achieve this using a techinique called expectation-maximisation [36, EM]: EM alternates between two steps. In the expectation step, the integral over the latents in Equation 2.2 is computed by calculating the expectation of the log likelihood over the latents: i.e. given Equation 2.4 $\implies$ $r_t \sim q_\theta(r_t) = p(r_0)\prod_{t'=0}^{t-1} q_\theta(r_{t'+1}|r_{t'})$:

$$\mathcal{L}(\phi, \theta, \mathcal{C}) = \sum_t \mathbb{E}_{r_t \sim q_\theta(r_t)} \log q_\phi(y_t|x_t, z_t, r_t, C) \tag{2.5}$$

Note that this expectation is a Monte Carlo approximation to the integral in Equation 2.2. In the maximisation step, the log likelihood $\mathcal{L}(\phi, \theta, C)$ is maximised with respect to the the parameters ϕ, θ, C.

2.1.3 Unsupervised models

In unsupervised learning, there are no well-defined x, y pairs to learn a mapping, or well-defined loss between network outputs and targets, or both. Clustering, reinforcement learning and neural density estimation [20] are all unsupervised learning problems.

2.1.4 Density estimation

Density estimation refers to a set of methods that attempt to approximate the probability density of some observed data, given only samples from the dataset. These can be non-parametric i.e., building histograms or kernel density estimation [135, 153], or parametric i.e., variational inference [21]. For the purposes of this thesis, we focus on variational inference.

Variational inference

Variational inference refers to a set of approaches that approximate the probability density over data $p(y)$, that are otherwise difficult to obtain in closed form. These approaches hypothesise a family of density functions $q_\phi(y)$, learn the density $q_{\phi^*}(y)$ within this family, with the minimum "distance" to the true density $p(y)$. These are models that attempt to learn the *distribution* of the data, rather

than match individual samples. Thus, they typically, do not require labeled data – however, the distinction between supervised methods and unsupervised variational inference becomes less clear in the case of *conditional densities* $p(y|x)$, where "labels" or inputs x are conditioning variables for the densities.

Variational autoencoders

Variational Autoencoders (VAEs)[84] represent a particular approach to variational inference, which incorporates dimensionality reduction of the data to a latent manifold. VAEs consist of an encoder, which stochastically maps data y to latents z — $z \sim q_\phi(z|y)$, and a decoder network which maps the latents z back into data y — $y \sim q_\theta(y|z)$. The objective is to capture the density of the data $p(y)$:

$$p(y) \approx \int q_\theta(y|z)q_\phi(z)\ dz \tag{2.6}$$

by maximizing $\log p(y)$. Since $q_\phi(z) = \int q_\phi(z|y)p(y)\ dy$ is hard to compute, we instead maximize a lower bound to $\log p(y)$:

$$\begin{aligned}\log p(y) \approx & \log \int q(y,z)\ dz \\ = & \log \mathbb{E}_{q_\phi(z)} \frac{q(y,z)}{q_\phi(z)} \\ > & \mathbb{E}_{q_\phi(z)} \log \frac{q(y,z)}{q_\phi(z)} \\ = & -\mathrm{D}_{KL}(q_\phi(z)\ ||\ q(y,z))\end{aligned} \tag{2.7}$$

This Kullback-Leibler divergence is called the evidence lower bound, and thus, the approach is called Evidence Lower Bound Optimization (ELBO). Note that when we wish to model data distributions for inputs x, all densities involved become conditioned on x i.e $q(\cdot|x)$.

GANs also perform variational inference, and can be used to approximate densities. They are also trained in an unsupervised manner. We expand more on GANs, and their use as generative models, in the following section.

2.2 Overview of generative adversarial networks

GANs have become the machine learning approach of choice for learning powerful models in a wide range of fields. GANs, like most machine learning approaches, are immensely flexible in that they can be adapted for generating a wide variety of data. They are used in widely disparate fields such as computer vision [125, 185, 189], medicine [12, 187, 199] and music [42]. Some of the well-known GAN variants are image-generating GANs: StyleGAN [80], which generates realistic images of people who never existed; Pix2Pix [74] and CycleGAN[200] which perform image-to-image translation; StackGAN [198] which performs text-to-image translation; BigGAN [25] which is the current state-of-the-art method for generating images from the ImageNet dataset [37]. This extensive application of GANs is enabled by the fact that the method is unsupervised (i.e. it does not require an explicit objective to optimize the parameters of the network)[1], completely

[1]For more details on supervised versus unsupervised learning, see section 2.1

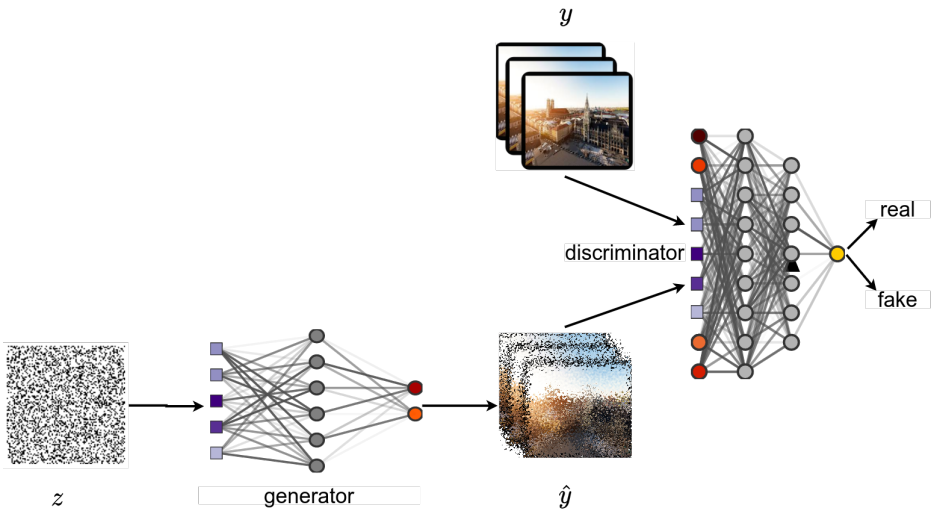

Figure 2.1: Generative adversarial networks. The generator takes latents z as input to generate outputs $\hat{y}$. The discriminator classifies generated $\hat{y}$ as fake, and targets y as real. The two networks are trained as adversarially.

agnostic to the nature of the target data and therefore easily scalable with the dimensionality of the data, and does not impose many restrictions on the architecture of networks used. This confers advantages to the GAN approach that few other machine learning approaches share.

2.2.1 Formulation

In the original formulation [50], GANs consist of two networks: a generator g_ϕ and a discriminator D_ψ. The objective of GANs training is to obtain a generator that produces random variables $\hat{y}$ that match target samples y from a probability distribution $p(y)$: in other words, the generator implicitly represents a distribution $q_\phi(\hat{y})$ that we want to match to $p(y)$. The generator network g_ϕ, parameterized by ϕ implicitly represents a distribution since it takes input noise $z \sim p(z)$ and transforms it into the target data $\hat{y}$: $\hat{y} = g_\phi(z)$.

$$\left.\begin{array}{l} z \sim p(z) \\ \hat{y} = g_\phi(z) \end{array}\right\} \hat{y} \sim q_\phi(\hat{y}) \tag{2.8}$$

The discriminator network D_ψ, parameterized by ψ, is a classifier: it classifies data from the target distribution $y \sim p(y)$ as "real" and generated data $\hat{y}$ as "fake". The two networks are trained adversarially: the generator is updated such that it fools the discriminator into classifying $\hat{y}$ as real, and the discriminator is trained to improve its classification accuracy. This is achieved using a minimax objective:

$$\min_\phi \max_\psi \mathcal{L}(\phi, \psi) = \mathbb{E}_{p(y)} \log \mathrm{D}_\psi(y) + \mathbb{E}_{q_\phi(\hat{y})} \log \left(1 - \mathrm{D}_\psi(\hat{y})\right) \tag{2.9}$$

The first term is the expectation of the discriminator classifying $y \sim p(y)$ as real, and the second term is the expectation that the discriminator classifies generated data $\hat{y} \sim q_\phi(\hat{y})$ Assuming that the generator and discriminator architecture have the capacity to fit the data y, the two networks will have converged to the Nash equilibrium [122], where the parameters of neither network ϕ, ψ

can be updated without resulting in a non-optimal value of the objective function in Equation 2.9. This is in contrast to most other machine learning methods, where the objective is to find the parameters of the neural network that results in the most optimal value for the objective function.

Nevertheless, Goodfellow et al.(2014) [50] show in Proposition 1., that if the discriminator is a perfect classifier then the corresponding generator minimizes the Jensen-Shannon Divergence (JSD) between the generated and target data distributions: i.e.,

$$\begin{aligned} \mathrm{D}^*_\psi(y) &= \max_\psi \mathcal{L}(\phi,\psi) = \max_\psi \mathbb{E}_{p(y)} \log\ \mathrm{D}_\psi(y) + \mathbb{E}_{q_\phi(\hat{y})} \log\ \left(1 - \mathrm{D}_\psi(\hat{y})\right) \\ &= \frac{p(y)}{p(y) + q_\phi(y)} \end{aligned} \tag{2.10}$$

$$\begin{aligned} \implies g^*_\phi &= \min_\phi \mathbb{E}_{p(y)} \log\ \mathrm{D}^*_\psi(y) + \mathbb{E}_{q_\phi(\hat{y})} \log\ \left(1 - \mathrm{D}^*_\psi(\hat{y})\right) \\ &= \min_\phi \mathbb{E}_{p(y)} \log\ \frac{p(y)}{p(y) + q_\phi(y)} + \mathbb{E}_{q_\phi(y)} \log\ \frac{q_\phi(y)}{p(y) + q_\phi(y)} \\ &= \min_\phi\ \mathrm{JSD}(p(y)||q_\phi(y)) \end{aligned} \tag{2.11}$$

The JSD is an objective function that reaches its optimal value (i.e. 0), when $q_\phi(y) = p(y)$. Thus, at the theoretical limit, the generator network implicitly represents the true data distribution $p(y)$, and forward passes through the generator are equivalent to sampling from $p(y)$. Thus, GANs are implicit density estimators.

Note that the generator transforms latents z *deterministically* into targets $\hat{y}$: hence, the choice of latent distribution and dimensionality are unrestricted, contributing the flexibility of the GAN framework. Furthermore, the discriminator as an *implicit loss* between the generated and target data, thus allowing a lot of leeway in the choice and dimensionality of the target data – it would otherwise be hard to define a good loss function between images or audio data for supervised training. Finally, since the discriminator and generator are both trained only with samples y from the distribution $p(y)$, we do not need to be able to evaluate $p(y)$ – this contributes to the flexibility and the scalability of the GAN framework. Similarly, we also do not need to evaluate $q_\phi(\hat{y})$ while training the GAN, thus allowing for freedom in the choice of architecture for the two networks. However, it is still possible to introduce inductive biases and domain knowledge via architecture and hyperparameter tuning e.g. using convolutional networks or recurrent neural networks when y is a time-series, or using Fourier-transforms of y when it has periodic structure.

2.2.2 Conditional generative adversarial networks

We have considered GANs that map noise z to targets y, implicity representing a distribution $p(y)$. This is equivalent to a distribution over images of cats, or all possible temperature values, or all possible words in the English language. However, in many real world applications, we are interested in distributions of target variables y, given a conditioning variable m i.e conditional distributions $p(y|m)$. This is equivalent to distributions over images of cats (y) of a particular breed (m), or the distribution of temperature values during different seasons, or the distribution of English words used in specific countries. GANs can be used as implicit density estimators of conditional densities [113]: e.g. for generating images given line-drawings [74], images given text [150], audio given text [40], etc. Practically, the generator and discriminator additionally take m as input (Figure 2.2),

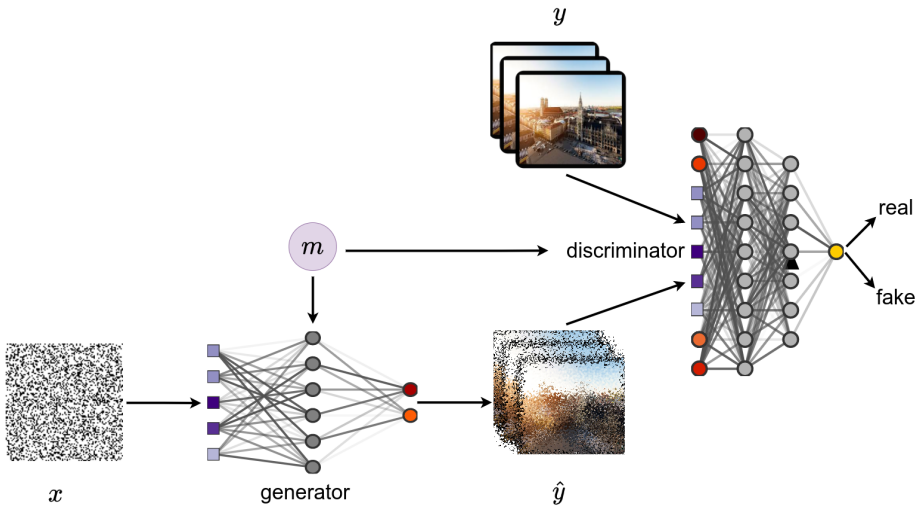

Figure 2.2: Conditional generative adversarial networks. Both the generator and the discriminator can additionally be conditioned on variable m, to obtain a conditional distribution over the data $p(y|m)$.

and theoretically the minmax objective function from Equation 2.9 is modified as follows:

$$\min_{\phi} \max_{\psi} \mathcal{L}(\phi, \psi) = \mathbb{E}_{p(y)} \log \ \mathrm{D}_{\psi}(y; m) + \mathbb{E}_{p(z)} \log \ \Big(1 - \mathrm{D}_{\psi}\left(g_{\phi}(z; m); m\right)\Big) \tag{2.12}$$

Finally, the generator implicitly represents the conditional density $p(y|m)$ at convergence. Conditional GANs are as flexible as the vanilla formulation in terms of dataset and network architecture, but do require labeled target data for training.

2.2.3 Challenges with GAN training

The flipside of the flexibility conferred by the GAN formulation, is enormous difficulty in training. The fact that the loss landscape for the generator changes with every update to the discriminator, makes the setup very sensitive to initial conditions and prone to mode collapse – where the GAN simply reproduces a single or a subset of samples from the target distribution to fool the discriminator. Moreover, the flexibility in architecture choice can also be an obstacle: an enormous amount of hyperparameter tuning is required in order to achieve stable training and convergence of the networks, and the corresponding computational cost is staggering. Myriad modifications to the GAN formulation have been suggested to mitigate issues of mode collapse and training stability, including changes to the minmax objective [6, 94, 108, 118], specific architecture choices [144, 197], penalising discriminator updates [56, 114] and general rules of thumb for training [99, 156]. However, new GAN variants emerge everyday, and the wheel must be reinvented (or at least existing solutions need to be mixed in different permutations and combinations) depending on the application.

In the following chapters, we show that the flexibility and scalability of GANs in learning conditional densities, and the fact that they require access to the target densities only via samples, are advantages that can be leveraged for modelling a wide variety of data: from the activity of neural populations to ocean depth profiles. We also demonstrate how the trained GANs can then be used to gain scientific insight into these domains.

Chapter 3

Adversarial training of neural encoding models on population spike trains

In this chapter, we provide some background and describe the contributions of the publication Ramesh et. al (2019) [145].

3.1 Background: neural encoding models

Understanding how information about input stimuli are encoded in neural population activity is an important problem in systems neuroscience. Populations of neurons spike in response to input stimuli, and theories about how the responses are influenced by different features of the stimulus abound – when expressed as statistical or dynamical systems models, they are collectively termed neural encoding models.

Hypotheses for neural encoding models include (but are not limited to) rate coding [3, 136, 165], where the stimulus influences the responses of individual neurons averaged over time or repeated stimulus presentations [1]; temporal coding [59, 151, 171, 173] where the precise timing of the spikes are thought to carry information about the stimulus; and phase coding [26, 81, 116] where neurons spike at different phases of an oscillatory input. These theories address how information about the stimulus is translated into spikes, and could potentially provide insight into how the brain parses information about the external world. Furthermore, understanding how stimulus features are encoded in neural activity is crucial for studying decision-making and behaviour – this has ramifications for a wide range of fields including medicine, artificial intelligence, and economics.

Sensory systems research typically investigates stimulus-response relationships. However, neural population activity is driven not just by an input stimulus, but also by latent processes that we cannot observe. For example, the stimulus could increase the average number of spikes (i.e., the mean firing rate) from two neurons independently (Figure 3.2). On the other hand, a latent process (e.g., synaptic connections) could cause one neuron to fire in tandem with another, independent of any input stimulus to the neurons. In both instances, the spike trains of the two neurons would be positively correlated. When both these mechanisms are at play, neural population activity is highly variable: a combination of stimulus-driven variability and stimulus-independent variability (Figure 3.1). Understanding how neurons encode stimuli requires disentangling these sources of

[1] This average response is called the firing rate. It has no biological significance, since neurons communicate via spikes, and not rates

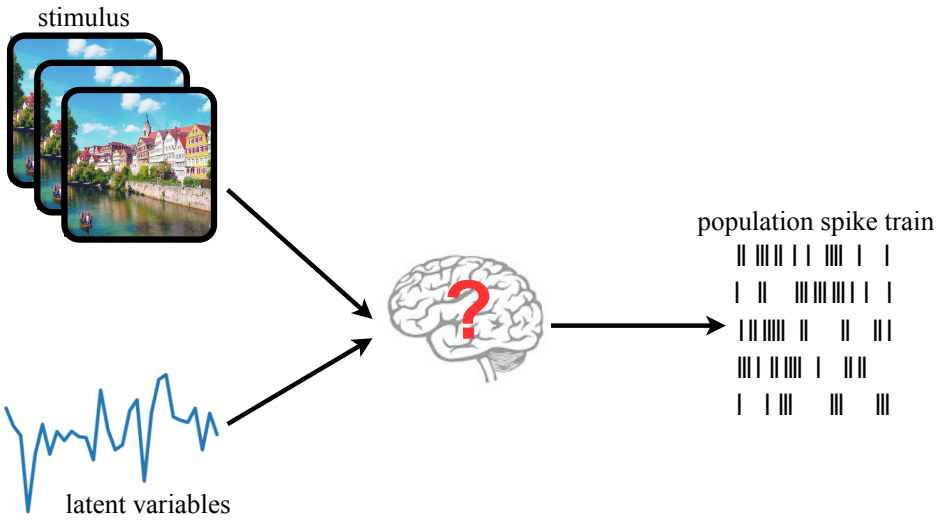

Figure 3.1: Neural encoding models for capturing signal and noise.

variability. Thus, in addition to stimulus-coding theories, sophisticated neural encoding models include hypotheses for latent processes driving neural activity: e.g., predictive coding [32, 44, 148], feedback from downstream brain regions [10, 159], attention modulation [51, 70, 100, 170], and adaptation [4, 18, 119, 180]. Testing these hypotheses can be hard in vivo, particularly when it involves multiple repetitions of experiments, with many variations or perturbations to the input stimulus, in order to tease apart the effect of stimuli and latent "noisy" processes.

3.1.1 Generative models of neural activity

Generative models for neural activity use hypotheses about stimulus and latent noise encoding to map stimuli to neural responses. These models are trained to generate "realistic" spike trains corresponding to input stimuli, by fitting them to experimentally measured spike trains from neural populations presented with the same stimuli.

The ideal generative model would capture not just the signal-driven component of population activity, but also the latent noise correlations that are independent of the stimulus (Figure 3.2).

Such generative models of neural activity can enable rapid hypothesis testing: having a model that perfectly reproduces the neural activity in response to a given stimulus would allow us to perform rapid in-silico experiments, and fine tune hypotheses that can then be tested experimentally. Furthermore generative models might enable experiments that are impossible in vivo: for example, with generative models that explicitly represent theories of latent noise processes influencing the population activity, we might be able to probe how input stimuli perturb the shared latent noise structure and connectivity between neurons.

3.1.2 Models for neural data

In this section, we discuss different formulations for generative models of neural population activity. We focus our discussion on statistical models: i.e. models with parameters ϕ that assume population spike trains y are samples from a distribution $q_\phi(y|x)$, where x denotes inputs to the neural

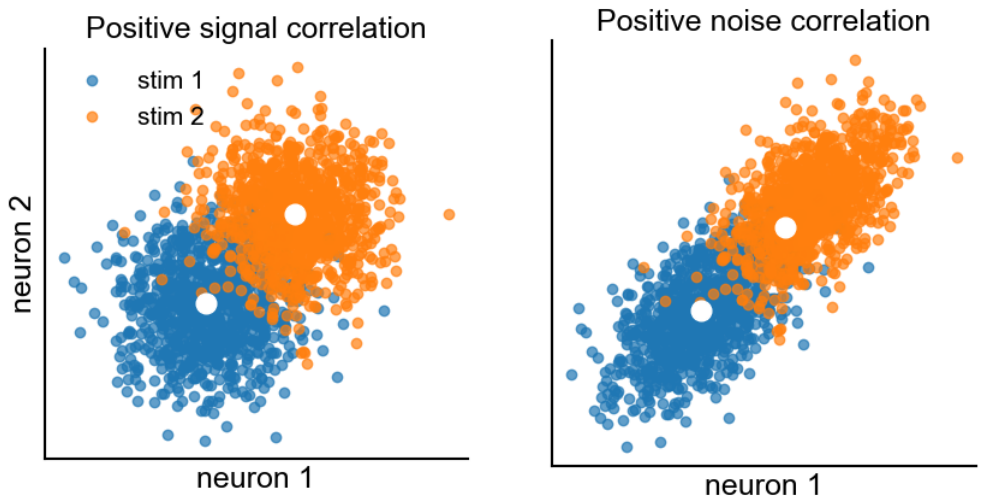

Figure 3.2: Signal and noise-driven correlations between 2 neurons. (Left) The activity of two neurons can be positively correlated only because their mean response (white) increases with increase in the amplitude of input stimuli ((blue and orange)). (Right) Neural responses can additionally be correlated because of latent noise processes unrelated to the stimulus.

population, and discuss their limitations. For a detailed description of statistical models, and related machine learning concepts, see section 2.1.

Maximum likelihood estimation for neural data

Most state-of-the-art encoding models explicitly model only certain features of the population activity: [9, 52, 110, 124, 177] all capture the relationship between stimulus and firing rate of the neural population i.e. spike trains averaged across multiple presentations of the same stimulus. These models are typically fit using maximum likelihood-based approaches: Given target spike trains y, conditioned on stimuli x, and a stochastic generative model of the data $\hat{y} = g_\phi(x, z), z \sim p(z) \implies \hat{y} \sim q_\phi(y|x)$, we learn the parameters ϕ of the generative model by maximizing the log likelihood of the target data y under $q_\phi(y|x)$, as described in Equation 2.1. These methods are useful and powerful, but typically only capture the mean and the variance of the population spike trains, while ignoring other features of the population activity. This is because MLE is guaranteed to be convex, and converge to a global optimum, only when the model assumes that the data distribution belongs to the exponential family, and ϕ parameterizes specific moments of the distribution [124]. MLE does not allow for modelling latent processes that contribute to noise correlations between neurons.

Latent variable models

Latent variable models [105, 193] endeavour to capture mean firing rate, pairwise correlations as well as higher order correlations in population activity, by explicitly incorporating a latent noise process into the generative model.This requires computing an integral over the latent variable r (Equation 2.2). Latent variable models, nevertheless approximate this integral to perform MLE for generative models with latent dynamics. We here focus on two examples: one using factor analysis, and one using a latent dynamical system.

Yu et al. (2008) [193] use factor analysis to learn a low-dimensional (i.e., latent) representation r of the population activity y, and a Gaussian process[149] to approximate correlations between the latents r across time. Under this model, the assumption is that the neural population activity has low-dimensional dynamics and the spikes at each time point are instantaneous high-dimensional projections (parametrized by C, with bias d) of the latent variables:

$$y_t = Cr_t + d + \epsilon,\ \epsilon \sim \mathcal{N}(0, \Sigma) \tag{3.1}$$

Note that Yu et al. (2008) [193] use a square-root transform of the spike counts in order to approximate the distribution of spikes as a Gaussian distribution as in Equation 3.1. They assume the latents at each time point are correlated via a Gaussian process with a square-exponential kernel $K(t, t') = \sigma \exp(-\frac{1}{\tau}(t - t')^2)$:

$$r_{1\ldots T} \sim \mathcal{N}(0, K) \tag{3.2}$$

Macke et al. (2011) [105] go one step further and also learn the distribution of the latents, rather than approximating it with a Gaussian process. They model a linear dynamical system for time-varying latents r_t with autoregressive weights A and bias b

$$r_{t+1} = Ar_t + b + \epsilon,\ \epsilon \sim \mathcal{N}(0, \Sigma) \tag{3.3}$$

Both models have well-defined likelihoods over y_t (Gaussian for Yu et al. (2008) [193] and Poisson for Macke et al. (2011) [105]), conditioned both on the inputs x_t and the latents r_t. The parameters of these models are learnt using expectation-maximisation (EM).

While latent variable models in combination with EM allow us to train effective generative models of neural data, capable of reproducing variability more accurately, they are computationally expensive, do not scale well for longer (in time) spike trains, and are limited by the requirement of having evaluable distributions for the latents and for the spiking data (e.g. Gaussian or Poisson distribution) that are easy to integrate over.

Copula-based models

Macke et al. (2009) [104] use copula-based models for capturing instantaneous mean and pairwise noise correlations of the neural population activity. This approach estimates a Gaussian distribution with an instantaneous mean and covariance matrix, such that when samples from this distribution are binarized, the resulting "spikes" have the same mean firing rate and pairwise correlations as the target neural population data: Given a spike train y with mean firing rate μ_y and covariance Σ_y, we define a Gaussian random variable $z \sim \mathcal{N}(\mu_z, \Sigma_z)$ such that

$$\begin{aligned}
\mu_y^i &= \Phi(\mu_z^i) \\
\Sigma_y^{ii} &= \Phi(\mu_z^i)\Phi(-\mu_z^i) \\
\Sigma_y^{ij} &= \Phi_2(\mu_z^i, \mu_z^j, \Sigma_z^{ij}) - \Phi(\mu_z^i)\Phi(\mu_z^i)
\end{aligned}$$

where i and j index the neurons in the population, Φ denotes the univariate standard Gaussian cumulative density function (CDF) and Φ_2 denotes the bivariate Gaussian CDF. With this model, when we sample $\mathcal{N}(\mu_z, \Sigma_z)$ and binarize them i.e $y_i = 1$ if $z_i > 0$, we obtain spike trains with mean μ_y and covariance Σ_y. Like latent variable models, this approach captures more statistics of the population activity than the vanilla likelihood-based approach, and is capable of perfectly

reproducing the pairwise correlational structure of neural activity. Its advantage over latent variable models stems from the fact that it does not require any integration step: it solves a constrained optimisation problem to estimate μ_x and Σ_x. However, it does not capture any other statistics of the data, cannot generalise to test data and does not explicitly use any information about the input x while generating spike trains.

Variational Autoencoders

The VAEs allows us to incorporate latent processes in modelling neural population activity via the encoder, and generate spike trains using the decoder. Pandarinath et al. [130] employ VAEs to capture spiking data, by using an encoder that represents a dynamical equation, similar to the latent dynamical systems models. However, since the ELBO objective involves computing the D_{KL} over the joint distribution of the latents and the data, there is no explicit marginalisation step over the latents z as with EM. However, since the latents are modelled by a dynamical system, computing the product (or sum, in log space) of the latent distribution per timestep is still a requirement.

In general, this is a more robust approach than supervised or copula models for modelling neural population dynamics. However, since the ELBO computation requires evaluable densities, it severely limits the space of possible latent distributions, or the architecture of the encoder network.

We now show that GANs can be used as a generative model of neural data, while eliminating many of the problems that we outlined with likelihood-based, latent variable, copula-based and variational autoencoder models.

3.2 Generative adversarial networks as neural encoding models

GANs are also capable of matching the density of observed samples via the generator network. Since they take latent noise variables as input in addition to conditioning variables, they are capable of capturing higher order correlations in data unlike MLE-based models. They can be set up to use information from inputs to the neural population via conditioning unlike the copula models. Finally, training them does not require integration over latent densities like the latent variable models. These advantages translate into greater flexibility in the choice of architecture, dimensionality and complexity of inputs and latent variables when modelling neural activity using GANs.

Although they come with their own pitfalls (section 2.2 for details), there exist several GANs-based generative models for neural data: e.g. Arakaki et al. [5] train GANs as rate models of neural population data, Molano-Mazon et al. [115] train GANs for spontaneous neural population spike trains. However, generating spiking data and capturing higher order correlations along with the stimulus-driven mean and variance of the population activity, is non-trivial. This would require directly matching binary spike trains from the generator to measured spike trains. If the generator is to produce spike trains , then it must incorporate a step function in its hidden layers. However, the step function would make it impossible to backpropagate gradients through the generator, and thus prohibit training for the GANs.

We would thus need solutions for backpropagating gradients through the generator network in order to train them: we would need surrogate gradient methods.

3.2.1 Surrogate gradients

Surrogate gradient methods are widely used for backpropagating gradients in spiking neural networks [123], for applications in natural language processing [89] where data can also be discrete, and optimisation for black-box models [53]. In general, surrogate gradient approaches aim to replace the intractable step in the forward pass (while generating data) or backward pass (while calculating gradients) with a tractable function.

Bengio et al. [16] use "straight-through" gradient estimation, where the gradient before the threshold function in the backward pass is carried through to the units after the threshold function, as if the threshold were the identity function. This works well, but leads to biased gradients. Neftci et al.[123], Maddison et al. [106] and Kusner et al. [89] all replace the threshold function with a smooth function, (e.g., sigmoid $\sigma(\cdot) = \frac{1}{1+\exp(-\cdot)}$) that closely approximates the threshold function, but produces continuous-valued outputs. This is known as concrete relaxation, and it also works well for computing gradients for generators of discrete data: Molano-Mazon et al. [115] use precisely this method to train a GAN for generating spontaneous spiking activity. However, this method also leads to biased outputs – the concrete relaxation tends to shift the mean firing rate of the neurons, because it allows for non-binary values of "spikes" to count towards the firing rate.

Another approach is to use the log-derivative trick [186]: if the loss is computed as an expectation of some function f over the data y, then the loss gradient can be expressed as an expectation over the derivative of the data distribution $q_\phi(y)$, without having to compute the derivative of the function f. Given a loss function $\mathcal{L}(\phi) = \mathbb{E}_{y\sim q_\phi(y)} f(y)$ for the parameters of the generative model ϕ, and assuming that the model outputs y are discrete random variables i.e $y \sim q_\phi(y)$, we compute the gradients as follows:

$$\frac{\partial}{\partial\phi}\mathcal{L}(\phi) = \mathbb{E}_{y\sim q_\phi(y)}\Big[f(y)\frac{\partial}{\partial\phi}\log\ q_\phi(y)\Big] \tag{3.4}$$

This method produces gradients that are unbiased, but with high variance. We can however mitigate the variance in gradients by using control variates [92]: terms added to the loss function, such that the gradients have the same mean as in Equation 3.4, but with lower variance. One option is to use a constant (with respect to ϕ): $\mathcal{L}'(\phi) = f(y) - c, \ni c = \min_c \mathrm{Var}(\frac{\partial}{\partial\phi}\mathcal{L}(\phi))$. Another approach, REBAR [178] computes the control variate by using concrete-relaxed estimates of the data y:

$$\begin{aligned}\frac{\partial}{\partial\phi}\mathcal{L}(\phi) =& \mathbb{E}_{y\sim q_\phi(y)}\Big[\Big(f(y) - \eta f(y_{\text{relax}}|y_g)\Big)\frac{\partial}{\partial\phi}\log\ q_\phi(y_g)\Big] \\ &+ \eta\frac{\partial}{\partial\phi}\mathbb{E}_{y_{\text{relax}}\sim q_\phi(y_{\text{relax}})}[f(y_{\text{relax}})]\end{aligned} \tag{3.5}$$

This approach provides lower variance gradients compared to the log-derivative trick, but is more computationally expensive.

In Ramesh et al. (2019) [145], we motivate the use of GANs as neural encoding models. We then compare the merits of different surrogate gradient estimation methods, specifically concrete relaxation, the log derivate trick and REBAR, to train GANs for generating spike trains. We show how GANs outperform some of the methods we described in this chapter, by capturing more statistics of spike trains while also providing more flexibility as neural encoding models. Thus, they potentially expand the scope for neuroscientific insight from generative models of spike trains.

3.3 Publication: Adversarial training of neural encoding models on population spike trains

This paper was accepted for a talk at the workshop "Real Neurons & Hidden Units" at the 33rd Conference on Neural Information Processing Systems in 2019, and was published in the conference proceedings.

Neural population activity exhibits variability both due to external stimuli and due to unobserved latent noise factors (Figure 3.1). Generative models for reproducing neural data typically focus on capturing only the stimulus-driven component [9, 110, 141], or on some aspects of the latent noise-driven component such as pairwise correlations [104, 130]. Accurate generative models of neural data are important for studying the effect of stimulus features on neural activity, and would potentially speed up experiments and hypothesis testing for the factors that drive neurons' behaviour. However, the discrete nature of the spikes in neural population activity makes setting up and training generative models of this activity, while capturing all aspects of its variability, non-trivial. This is further compounded by the dimensionality of the data: it becomes harder and more computationally expensive to capture all statistics of the population activity with increasing number of neurons in the population, or the time for which the activity is recorded.

Generative adversarial networks (GANs) are good generative models for data of arbitrarily high dimensionality, such as images [74, 79] or audio [42]. They are capable of learning the distribution of a given dataset $p(y)$, based only on samples y from this distribution. This makes them the optimal tool for reproducing neural data. Arakaki et al.[5] and Molano-Mazon et al. [115] attempt to reproduce firing rates and spontaneous spiking activity respectively using GANs. However, in order to have a generative model to study stimulus effects on population activity, one would have to train a GAN to reproduce spikes given an input stimulus.

We propose to do this using conditional GANs (Figure 3.3). We use a generative network g_ϕ that takes both the stimulus x and latent noise $z \sim p(z)$ as input, and produces binary-valued spikes:

$$y \sim \text{Bernoulli}\left(g_\phi(z, x)\right),\ z \sim p(z) \tag{3.6}$$

We use a discriminator D_ψ conditioned on stimulus x, to differentiate between generated spikes and groundtruth or experimentally measured spikes. We train the two networks adversarially with a cross-entropy loss:

$$\phi^*, \psi^* = \min_\phi \max_\psi \mathbb{E}_{p(y|x)} \log \mathrm{D}_\psi(y|x) + \mathbb{E}_{p(y|g_\phi(x,z))} \log \left(1 - \mathrm{D}_\psi(y|x)\right) \tag{3.7}$$

The discrete nature of the spikes makes backpropagation of gradients impossible for the generator network g_ϕ parameters. We thus turn to surrogate gradient methods in order to calculate the gradients for ϕ – for a detailed discussion of surrogate gradient methods, see section 3.2. We tried 3 different methods of gradient estimation: first, concrete relaxation [106]. Given the discrete step function for obtaining Bernoulli random variables:

$$\left.\begin{aligned} u &\sim \mathcal{U}(0,1) \\ y &= H(u; 1-p) \end{aligned}\right\} \Rightarrow y \sim \text{Bernoulli}(p) \tag{3.8}$$

(where $\mathcal{U}$ denotes the uniform distribution, and $H(\cdot; h)$ denotes a step-function, with threshold at h), concrete relaxation replaces H with a smooth function i.e. sigmoid $\sigma(\cdot) = \frac{1}{1+\exp(-\frac{1}{\tau}\cdot)}$, where τ

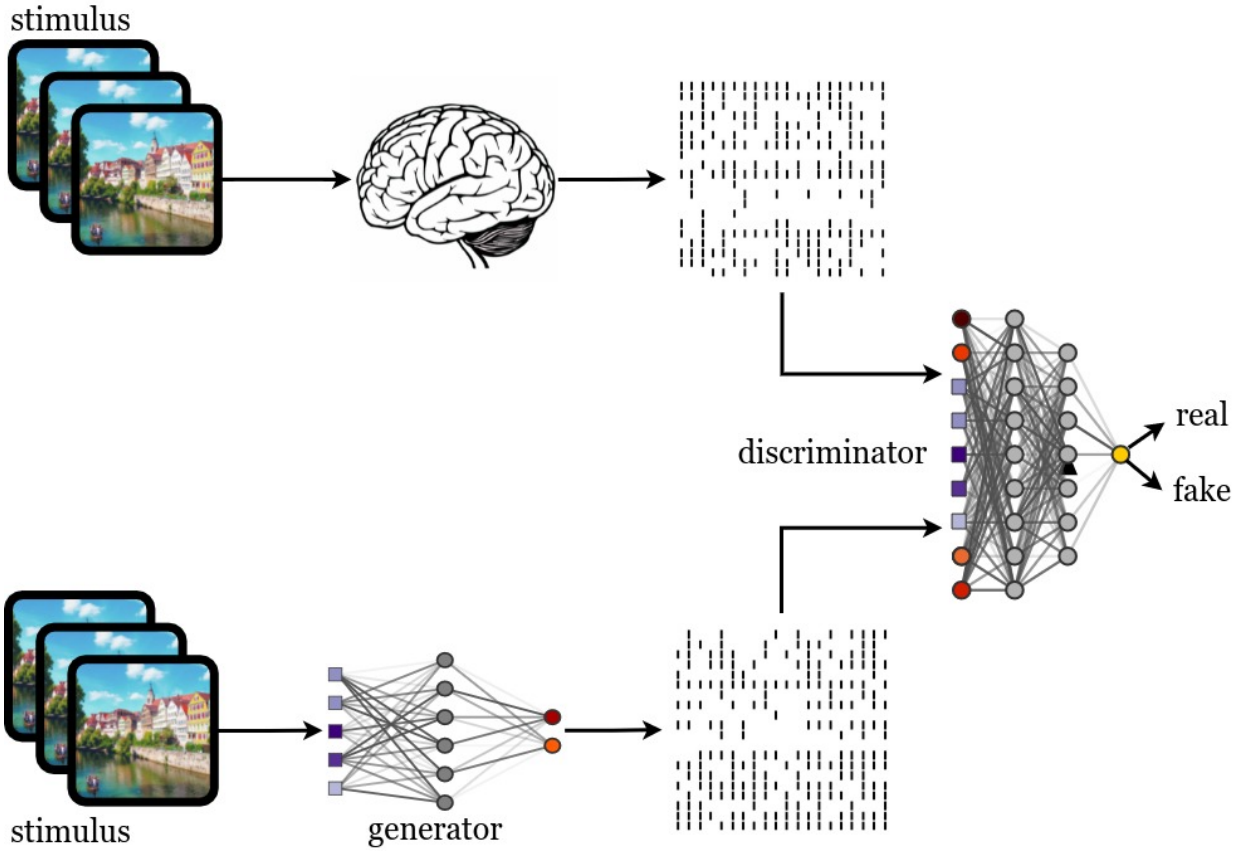

Figure 3.3: Generative adversarial networks for reproducing neural population activity.

controls the steepness of the sigmoid slope i.e $\lim_{\tau \longrightarrow 0} \sigma(\cdot) \longrightarrow H(\cdot)$. However, concrete relaxation is known to lead to biased outputs.

Thus, we also tried the log-derivative trick (also called the REINFORCE gradient estimator) [186]. Given that we can write down the likelihood $p(y|x,z)$ for a Bernoulli distribution in closed-form, and the loss function for the generator assuming a fixed discriminator D'_ψ:

$$\begin{aligned}\mathcal{L}(\phi) &= \mathbb{E}_{y \sim p(y|g_\phi(x,z))} \; \log\Big(1 - \mathrm{D}'_\psi(y|x)\Big) \\ &= \int dy \; p(y|g_\phi(x,z)) \log\Big(1 - \mathrm{D}'_\psi(y|x)\Big),\end{aligned} \tag{3.9}$$

we estimate the gradient for this loss function as follows:

$$\frac{d}{d\phi}\mathcal{L} \approx \int dy \; p(y|g_\phi(x,z)) \; \log\Big(1 - \mathrm{D}'_\psi(y|x)\Big) \; \frac{d}{d\phi} \log p(y|g_\phi(x,z)). \tag{3.10}$$

Finally, since the REINFORCE estimator is known to produce high-variance gradients, we also tried REBAR [178], which uses REINFORCE to estimate gradients. In addition to this, it uses concrete-relaxed spikes from the generator as control variates to reduce the variance of the REINFORCE gradients. Given

$$y_{\text{relax}} = \sigma(g_\phi(x, z, u)) \begin{cases} u \sim \mathcal{U}(0,1) \\ y_{\text{relax}} = \sigma(g_\phi(x, z) + \frac{u}{1-u}) \end{cases} \tag{3.11}$$

$$f(y|x) = \log\left(1 - \mathrm{D}'_\psi(y|x)\right), \tag{3.12}$$

the gradient is estimated as:

$$\frac{d}{d\phi}\mathcal{L} \approx \mathbb{E}_{p(y|g_\phi(x,z)),\, p(y_{\text{relax}}|y)}\left[\left(f(y|x) - \eta f(y_{\text{relax}}|x)\right)\frac{d}{d\phi}\log\ p(y|g_\phi(x,z))\right] + \eta\frac{d}{d\phi}\mathbb{E}_{p(y_{\text{relax}}|g_\phi(x,z))}[f(y_{\text{relax}}|x)]. \tag{3.13}$$

While training the GANs on simulated data from a Bernoulli distribution with three different gradient estimators, we found that concrete relaxation did indeed lead to biased outputs. We also showed that the bias stems from the mismatch between the binary-valued groundtruth activity, and the continuous-valued outputs from the generator: when we changed the groundtruth data so that it also consisted of continuous-valued "spikes", the bias in the outputs from the concrete-relaxed generator disappeared. We also found that there was no real difference in the output or the ability of the GANs to capture the statistics of the groundtruth data when using either REINFORCE or REBAR.

We then trained the GANs to reproduce data recorded from the primary visual cortex of a macaque [163], which was measured while the monkey watched a video. We showed that the GANs (irrespective of training with REBAR or REINFORCE) perfectly reproduced target data statistics including the mean, variance, pairwise correlations and the population spike histogram. This was in contrast to a maximum-likelihood based model, that only captured the mean and variance, and a copula-based model [104], which captured the mean, variance and the pairwise correlations (for detailed information on maximum likelihood estimation and copula-based models, see section 3.1). We also, found that the GANs and the maximum-likelihood model did not generalise to test data i.e. they did not accurately reproduce neural activity for video frames that they did not have access to during training. However, we found that artificially increasing the signal-to-noise-ratio of the target data improved generalization, indicating that poor test performance on the original data was due to insufficient information about the stimulus-driven component of the activity in the recorded data. We also found that GANs were the only models of the three capable of reproducing activity with heteroscedastic latent factors: i.e., the variance of the latent factors is stimulus-dependent. Finally, we showed that the stimulus features that contribute to the GANs capturing higher order correlations were completely different from the stimulus features that drive the maximum-likelihood model's estimate of the population activity. This indicates that capturing all aspects of the neural population data might lead to different insights about the system's behaviour in response to input, compared to a more restricted model.

To summarize, we showed how GANs can be used as generative models of discrete neural activity, using surrogate gradients. We demonstrated their capacity to capture more aspects of the neural population variability than existing models, and the utility of fitting such models for understanding stimulus-response relationships.

Author contributions

The paper was co-authored with Mohamad Atayi and Jakob H. Macke.

Jakob H. Macke had the idea for using GANs and surrogate gradients for reproducing neural data. Mohamad Atayi implemented the code for using surrogate gradients on the simulated Bernoulli data. I implemented code for applying GANs, maximum-likelihood estimation and copula-based models to the macaque primary visual cortex data, and generated all results and figures for the paper. Jakob H. Macke and I wrote the paper.

3.4 Summary

In this chapter, we have discussed the different models typically used for generating neural data. We argued for the uses of good generative models of data. We also discussed how unsupervised learning, and GANs in particular, may be used for generating realistic population activity, and how we can use surrogate gradients to backpropagate gradients for spiking networks.

In the publication Ramesh et al. (2019) [145], we demonstrate how to construct GANs for generating population spike trains given an input stimulus, compute gradients for the GAN generator using some of the surrogate gradient methods described above and the findings from training such GANs.

Chapter 4

GATSBI: Generative Adversarial Training for Simulation-Based Inference

In this chapter, we provide some background and describe the contributions of the publication Ramesh et al. (2022) [147].

4.1 Background: Bayesian inference for scientific simulators

Various disciplines in science model study real-world phenomena by modeling them with reduced representation: simulators. Simulators are typically amenable to hypothesis testing, and have parameters θ representing real-world conditions or experimentally measurable quantities e.g. channel conductances in the Hodgkin-Huxley model of action potentials in neurons [66]; the gravitational constant in Newtonian physics; rate constants in chemical kinetics reactions. The outputs of the simulator x usually embody real-world observations such as neural population activity, planetary orbits, or chemical compositions of mixed liquids. When these simulators are stochastic i.e., the simulator incorporates noise to account for unobserved or latent processes influencing the real-world phenomena they model, the corresponding outputs x can be thought of as draws from a likelihood distribution, conditioned on the simulator parameters θ: $x \sim p(x|\theta)$.

4.1.1 Simulation-based inference

Hypothesis testing with stochastic simulators involves asking which sets of interpretable parameters θ in the simulator could give rise to a particular observation x_{o}. It also involves obtaining uncertainty estimates due to noise in the simulator e.g., asking whether the same parameter set could lead to vastly different observations, or if multiple parameters could give rise to the same observation. Bayesian inference would allow us to pursue these questions, by obtaining a posterior distribution over the parameters θ, given the observed data x i.e., $p(\theta|x)$. The posterior density then gives us insight into how the parameters influence the outputs of the simulator, and thus enables us to probe the simulator in ways that may not be possible without extensive perturbation or ablation experiments on the simulator. It may also reveal dependencies between the simulator parameters.

Given a prior $\pi(\theta)$ over the parameters that incorporates any domain knowledge about the

simulator, and the likelihood of the simulator $p(x|\theta)$, Bayes' Rule allows us to obtain the posterior:

$$p(\theta|x) = \frac{p(x|\theta)\ \pi(\theta)}{\int p(x|\theta)\ \pi(\theta)\ d\theta} \tag{4.1}$$

The posterior is relatively easy to compute when we know the simulator likelihood. However, for most simulators, particularly mechanistic ones, the likelihood is intractable – we cannot evaluate it either because it is computationally expensive to calculate, or the internal computations of the simulator are inaccessible. For such 'black-box' simulators, the set of approaches that perform Bayesian inference are collectively termed Simulation-Based Inference (SBI) [35], since these methods use simulations or samples from running the simulator forward in lieu of the likelihood $p(x|\theta)$.

4.1.2 Approximate Bayesian Computation

Classical methods for SBI are usually collectively termed Approximate Bayesian Computation (ABC) [162]. ABC approaches typically work on the principle of rejection sampling [142, 169]. In ABC, we sample $x \sim p(x|\theta)$ via forward passes through the simulator for multiple $\theta \sim \pi(\theta)$, and accept θs that produce xs sufficiently similar to x_o (Figure 4.1). In other words, we accept $\theta \sim \pi(\theta)$ if $\rho(x, x_o) < \epsilon$, where ρ is a similarity metric between the simulations x and observation x_o, and ϵ is a tolerance value. In the limit $\epsilon \to 0$, the accepted θs are samples from the posterior $p(\theta|x_o)$. However, in practice, smaller ϵ would make the set of θs that lead to $\rho(x, x_o) < \epsilon$ vanishingly small in the prior domain, and require a large number of samples from the simulator to obtain a posterior. However, larger ϵ values result in poorer approximations to the posterior. Thus there exists a trade-off between accuracy and computational cost for inference with ABC approaches. Accurate posterior estimation with ABC also becomes more problematic with increasing observation and parameter dimension.

Different ABC methods attempt to reduce the accuracy-computation trade-off, and improve efficiency of the rejection sampling approach described above [13, 23, 109, 176]. For example, Beaumont et al. (2002) [13] sequentially improve the proposal distribution for sampling θ, starting from the the prior $\pi(\theta)$, to converge to an accurate estimate of the posterior $p(\theta|x_o)$. The sequential approaches use ideas from sequential importance sampling [97, 174] and particle filtering [38]. However, inference in this setting does not *amortize* easily: the posterior approximated with this approach is only valid for a *particular* observation x_o. With every new observation, the procedure needs to be repeated again to get a different posterior.

4.1.3 Neural density estimation

An alternative set of classical approaches that do not rely on rejection sampling are the "synthetic likelihood" approaches. These methods attempt to approximate the intractable simulator likelihood $p(x|\theta)$ using density estimation (Figure 4.2) e.g., assuming that the likelihood is Gaussian [188], and estimating its mean and variance from simulations. These synthetic likelihoods are then combined with standard ABC methods like Markov Chain Monte Carlo (MCMC) to obtain posteriors. However, these methods also suffer from the curse of dimensionality and poor amortization e.g., we would need to perform MCMC anew for each new observation. Several problems with the classical approaches are alleviated by a second set of methods: conditional neural density estimation. These methods rely on neural networks to directly approximate either the likelihood or posterior. Papamakarios et al. [133] and Lueckmann et al.[103] estimate the likelihood $p(x|\theta)$ and

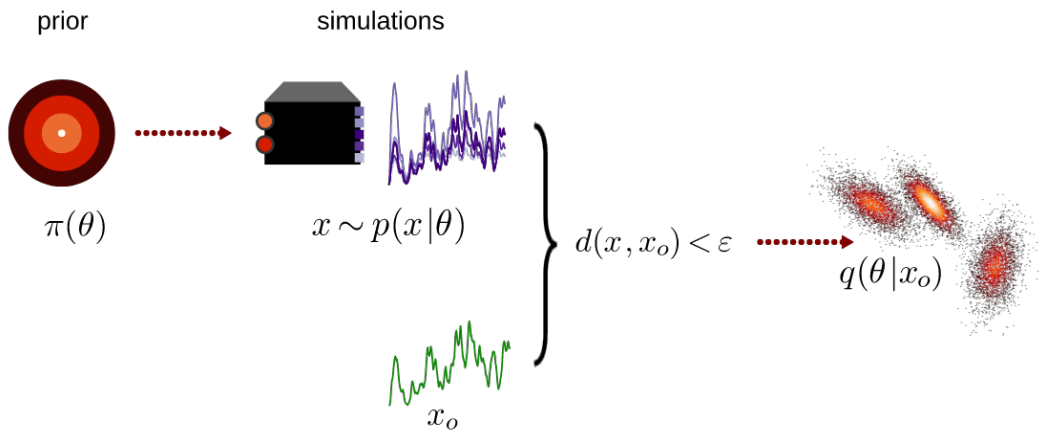

Figure 4.1: Approximate Bayesian Computation (ABC) approaches samples $\theta \sim \pi(\theta)$ and filter these samples via a distance d metric defined on the observations to approximate sampling from the posterior $p(\theta|x)$.

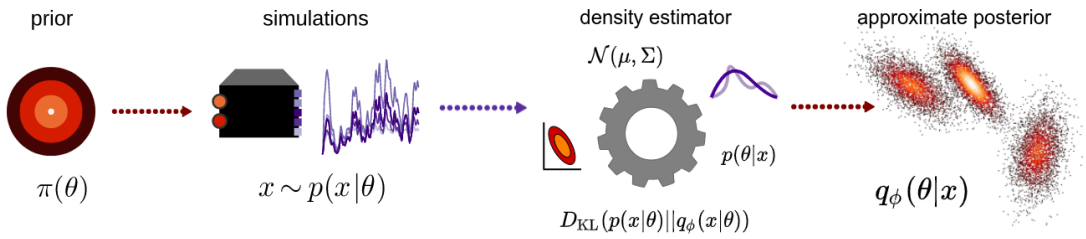

Figure 4.2: Neural density estimation approaches train density estimators using samples $\theta, x \sim \pi(\theta)p(x|\theta)$ to get an approximation to the posterior $q_\phi(\theta|x)$.

Hermans et al. [64], Izbicki et al. [75], Pham et al. [140] and Thomas et al. [172] all train ratios of densities e.g. $\frac{p(x|\theta')}{p(x|\theta)}$. The trained networks can be used to obtain samples from the posterior using MCMC sampling. Alternatively, many researchers [55, 101, 131, 143] train neural networks that directly estimate the posterior $p(\theta|x)$, given a particular observation. These methods variously use multi-layer perceptrons (MLP), mixtures of Gaussians, or normalizing flows [132] as the density estimator, which is trained on θ, x pairs sampled from the prior $\pi(\theta)$ and $p(x|\theta)$. These methods typically rely on the density estimators producing an evaluable posterior density [55, 103, 131], or on MCMC sampling to obtain samples from the posterior. Of these methods, those training density estimators to produce an evaluable posterior density can be used for amortized inference: the density estimator needs to be trained only once, and can easily be evaluated multiple times to obtain posteriors corresponding to different x_os. The methods estimating likelihoods or ratios of densities, on the other hand, require new MCMC-sampling chains for every observation x_o, which can become computationally expensive, particularly for high-dimensional parameters.

4.1.4 Sequential inference with neural density estimators

Neural density estimation is optimized for amortized inference. However, for many problem settings we are interested only in a posterior for a particular observation x_o: e.g., we are interested only in bursts of spikes from a neuron, or modeling planetary orbits in the aphelion phase. In these cases, and also when the simulator is computationally expensive, fine-tuning the density estimator to estimate posteriors for a particular observation is both scientifically more relevant and more sample-efficient. These fine-tuning approaches for neural density estimators are collectively called sequential inference.

In sequential inference, the density estimator is nudged to get better at approximating the posterior only for a particular observation, by training it on $\theta, x \sim \pi(\theta)p(x|\theta)$ pairs, such that the simulator samples x are similar to x_o. This is typically done by first training a rough approximation to the full posterior $q_\phi(\theta|x) \approx p(\theta|x)$, obtaining "proposal" θ samples from it by conditioning on x_o ($\theta' \sim q_\phi(\theta'|x_o)$), simulating x's from the simulator corresponding to θ', and using (θ', x') to continue training the density estimator. This means that we perform density estimation over several "rounds", and at each round we refine the posterior estimate by training it on observations x' clustered around x_o, and the parameters θ' that generate them. Over multiple rounds of the process, the density estimator would thus converge to $p(\theta|x_o)$.

Neural density estimators for likelihoods and ratios of densities allow for sequential inference, but always in conjunction with MCMC sampling, which can be computationally expensive and challenging for high-dimensional parameter spaces. Neural density estimators of the posterior, on the other hand, do not require MCMC sampling during sequential inference, and we focus the discussion of sequential inference to these estimators.

With these posterior estimators, in successive rounds of sequential inference, we do not sample θ's from the prior $\pi(\theta)$, but instead sample from a proposal distribution $\tilde{\pi}(\theta')$ that would in principle lead to a more accurate estimate of the posterior $p(\theta|x_o)$. Theoretically, the proposal distribution could take any form, but we usually set it to an estimate of the posterior i.e., $\tilde{\pi}(\theta') = q_\phi(\theta'|x_o)$. However, with the proposal distribution, the density estimator no longer represents to true Bayesian posterior $p(\theta|x_o)$, but rather a proposal posterior:

$$\tilde{p}(\theta'|x') \propto p(x|\theta')\tilde{\pi}(\theta'). \tag{4.2}$$

This nescessitates corrections to the learned density estimator to obtain the true posterior:

$$\begin{aligned} \frac{\tilde{p}(\theta|x)}{p(\theta|x)} &\propto \frac{p(x|\theta)\tilde{\pi}(\theta)}{p(x|\theta)\pi(\theta)} \\ \implies p(\theta|x) &\propto \tilde{p}(\theta|x)\frac{\pi(\theta)}{\tilde{p}(\theta)} \end{aligned} \tag{4.3}$$

Papamakarios et al. [131] approximate posteriors as mixture of Gaussians, and have Gaussian or uniform priors. Thus, they correct their learned proposal posteriors after training: this is relatively easy since the ratio in Equation 4.3 works out to be the ratio of the variances proposal and prior distribution. To enable the use of non-Gaussian priors, and non-Gaussian density estimators, Lueckmann et al [103] use the ratio in Equation 4.3 to correct the density estimator during training, by using the ratio as importance weights in the loss function:

$$\phi^* = \min_\phi \mathbb{E}_{\tilde{p}(\theta|x)} \frac{\pi(\theta)}{\tilde{p}(\theta)} \log q_\phi(\theta|x) \tag{4.4}$$

This loss allows $q_\phi(\theta|x)$ to converge to the true posterior $p(\theta|x)$ rather than the proposal posterior $\tilde{p}(\theta|x)$. Greenberg et al. [55] use the inverse ratio, also during training, to correct the density estimator directly, rather than the loss i.e. with $\tilde{q}_\phi(\theta|x) = \frac{\tilde{p}(\theta)}{\pi(\theta)} q_\phi(\theta|x)$:

$$\phi^* = \min_\phi \mathbb{E}_{\tilde{p}(\theta|x)} \log \tilde{q}_\phi(\theta|x) \tag{4.5}$$

At convergence, this would also result in the density estimator approximating the true posterior rather than the proposal posterior.

Thus, sequential inference allows for more accurate, sample efficient posterior density estimation. However, they also require further algorithmic choices and hand-tuning.

4.1.5 Drawbacks of approximate Bayesian computation and synthetic likelihood approaches

As discussed in the previous sections, classical ABC approaches rely on rejection sampling, and typically involve a trade-off between inference accuracy and computational cost. The efficiency of ABC methods with respect to this trade-off also suffers with increasing dimensionality of the parameters and observations.

Synthetic likelihood approaches also have some drawbacks. Density estimation approaches approximating the likelihood or a ratio of densities, rely on MCMC sampling, and thus, suffer from the same curse of dimensionality as as the ABC approaches. Methods directly estimating the posterior are not limited by MCMC sampling, but require density estimators that provide evaluable approximations to the posterior in order to compute the loss function during training.

4.1.6 Variational autoencoders and simulation-based inference

VAEs (described in subsection 2.1.4) also perform inference over latents z (equivalent to parameters θ in SBI). However, note that they are different from the density estimators described here in two key aspects: first, they require a parametrized decoder $p_\theta(x|z)$ whose parameters are also optimized via ELBO in Equation 2.7. The equivalent quantity in the SBI setting would be the simulator which is black-box and cannot be optimized. Second, even if we were to assume a decoder with fixed parameters that we do not update, we train the VAEs encoder (or equivalently, the SBI posterior density) with a reverse KL-divergence between the true and approximate posterior (Equation 2.7). In the SBI setting, this would require evaluating the true posterior $p(\theta|x)$ under samples from the approximate posterior $q_\phi(\theta|x)$, which is also impossible due to the black-box simulator.

4.2 Generative adversarial networks in simulation-based inference

GANs and SBI are dissimilar in many respects, and solve different density estimation problems. Nevertheless, several approaches leverage GANs for blackbox simulators: Kim et al. [82] and Louppe et al.[98] learn optimal priors, by training a discriminator to differentiate between simulated and observed samples. Jethava et al. [76] use two sets of generators and discriminators to simultaneously learn priors and summary statistics for the observations over which the prior distribution is optimized. Parikh et al. [134] train a conditional GAN to solve 'population of models' problems; Britton et al. [24], Lawson et al. [91] and Adler et al.[2] use Wasserstein-GANs [6] to solve Bayesian inverse problems in medical imaging.

In Ramesh et al. (2022) [147], we elucidate the connection between GANs in their original formulation and SBI, and show how GANs can be used as implicit posterior density estimators. We also show how the GANs used in this way, solve a similar density estimation problem as VAEs, and the advantages that this insight provides.

4.3 Publication – GATSBI: Generative Adversarial Training for Simulation Based Inference

This paper was presented as a poster at the The Tenth International Conference on Learning Representations in 2022 and published in the conference proceedings [147].

Scientific simulators for modeling real-world phenomena are common across many scientific fields. These simulators typically have interpretable parameters θ, with values tuned so that the model produces simulations x that are similar to experimental measurements x_{o}. These simulators are typically used to formulate hypotheses about the system they model. This endeavour, however, requires estimates of model uncertainty over the parameters i.e., calculation of the posterior density $p(\theta|x)$, which relies on some prior belief over parameters $\pi(\theta)$, and the likelihood of simulations from the simulator $p(x|\theta)$ (Equation 4.1). However, for most simulators, the likelihood is intractable, and accessible only via samples x from the simulator. The set of methods for obtaining Bayesian posteriors for simulators using only simulated samples x is called simulation-based inference (SBI).

Existing approaches for SBI include Approximate Bayesian Computation (ABC)[58], which typically involve a rejection sampling scheme from the prior $\theta \sim \pi(\theta)$, where θs are rejected based on a distance metric computed on the corresponding observations from the likelihood: $d(x, x_{\mathrm{o}}),\ x \sim p(x|\theta)$. However, these approaches do not scale well with the dimensionality of θ or x. There has been some work to develop ABC methods for high-dimensional parameter spaces, although under the assumption of Gaussian likelihood [127], or low-dimensional summary statistics [87, 152].

Another set of approaches is neural density estimation, where machine learning approaches are used to approximate either the likelihood [101, 133], the posterior [55, 103, 131] or ratios of densities that can be used for MCMC sampling from the posterior[41, 64]. These approaches typically train mixture density networks [19], Gaussian Processes [149] or flows [132]. However, these methods either rely on having evaluable estimates of the posterior density, or MCMC sampling from an evaluable density to get posterior samples. Both of these requirements do not scale with the dimensionality of the data.

While GANs share many ostensible similarities with SBI e.g. an intractable generative model of data that does not have a tractable density, we show that the two approaches are very different. As in SBI, GANs also only have access to the target density via samples, and solve a density estimation problem without a well-defined a loss function. However, GANs and SBI are dissimilar in several ways. First, GANs training is predicated on the generator being differentiable, unlike the simulator in SBI, which is assumed to be black-box in the most general formulation. Second, the latent variable inputs to GANs are always user-determined and therefore, almost always evaluable, unlike the latent noise in the SBI simulator contributing to its stochasticity. Finally, the parameters of the GANs are learnt as point estimates i.e., there is no explicit notion of a prior density or variability over parameters, unlike SBI in which the goals is to estimate a density over the parameters. However, GANs excel at density estimation in high-dimensional spaces, and thus could be leveraged to extend existing the range of possible applications for SBI. To that end, we introduce Generative Adversarial Training for Simulation-Based Inference (GATSBI). It is a GANs-based neural density estimation

approach for SBI. Assuming we have $(\theta, x) \sim \pi(\theta)p(x|\theta)$, GATSBI consists of a generator network

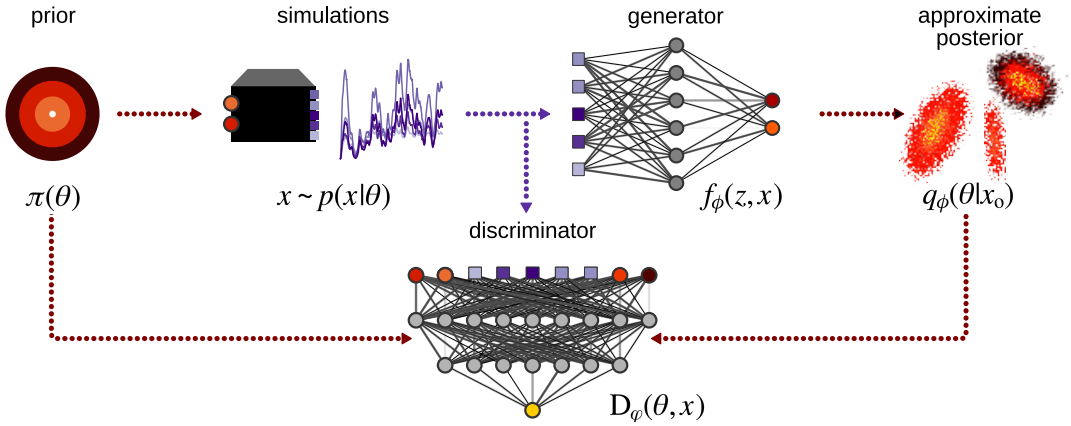

Figure 4.3: Generative adversarial training for simulation-based inference.

f_ϕ that takes latents $z \sim p(z)$ and x as inputs and generates θ – we thus implicitly sample from an approximate posterior $q_\phi(\theta|x)$ with a forward pass through the generator (Figure 4.3):

$$\left.\begin{array}{l} z \sim p(z) \\ \theta = f_\phi(z,x) \end{array}\right\} \Rightarrow \; \theta \sim q_\phi(\theta|x). \tag{4.6}$$

The discriminator network D_ψ differentiates between $\theta \sim \pi(\theta)$ and $\theta \sim q_\phi(\theta|x)$, conditioned on the corresponding simulations x. We show (in Proposition 1) that when the generator and discriminator are trained adversarially using the cross entropy loss:

$$\phi^*, \psi^* = \min_\phi \max_\psi \mathbb{E}_{\pi(\theta)p(x|\theta)} \log \mathrm{D}_\psi(\theta|x) + \mathbb{E}_{q_\phi(\theta|x)p(x)} \log\Big(1 - \mathrm{D}_\psi(\theta|x)\Big), \tag{4.7}$$

the generator converges to the the true posterior $p(\theta|x)$, by minimizing the Jensen-Shannon Divergence(JSD) $\mathrm{JSD}(p(\theta|x)||q_\phi(\theta|x))$, assuming an ideal discriminator.

We also show that GATSBI is an adversarial inference approach: it can be re-cast as a variational auto-encoder (VAEs) with a fixed decoder, where the ratio of densities in the KL-divergence term of the ELBO loss (Equation 2.7 are approximated with a discriminator. In Proposition 2, we show that the GATSBI generator (which minimizes the JSD), also minimizes a reverse KL-divergence. This is in contrast to existing neural density estimation approaches for SBI, which usually minimize the forward KL-divergence. Thus, GATSBI is an SBI method that optimises the reverse KL-divergence, rather than the forward KL-divergence, and can thus potentially capitalize on the attendant advantages of one objective over the other e.g., mode-seeking versus mode-covering behaviour.

This insight into GATSBI's training objective also allowed us to formulate a new method of sequential inference, that takes advantage of the reverse KL objective. In sequential inference, we attempt to correct our posterior estimate since we sample θs from a proposal prior $\tilde{\pi}(\theta)$, rather than the true prior $\pi(\theta)$. Thus, our estimate converges to a proposal posterior $\tilde{p}(\theta|x)$ rather than the true posterior $p(\theta|x)$. We can correct our estimate using importance weights $\frac{\pi(\theta)}{\tilde{\pi}(\theta)}$ in the training objective [103] for $q_\phi(\theta|x)$ (Equation 4.4) or correcting the forward pass through the density estimator using inverse importance weights [55] (Equation 4.5). Since GATSBI minimizes an objective using *samples* from the generator (Equation 4.7), we can correct the latent distribution

$p(z)$ of the generator *before* the forward pass, such that it generates samples from an approximate proposal posterior $\tilde{q}_\phi(\theta|x)$:

$$p_t(z) = p(z)\ (\omega(f_\phi(z,x)|x))^{-1} \tag{4.8}$$

$$\omega(\theta, x) = \frac{\pi(\theta)}{\tilde{\pi}(\theta)} \frac{\tilde{p}(x)}{p(x)} \tag{4.9}$$

$$\left.\begin{array}{l} z \sim p_t(z) \\ \theta = f_\phi(z,x) \end{array}\right\} \Rightarrow\ \theta \sim \tilde{q}_\phi(\theta|x) \tag{4.10}$$

$$\implies\ \phi^*, \psi^* = \min_\phi \max_\psi \mathbb{E}_{\tilde{\pi}(\theta)p(x|\theta)} \log \mathrm{D}_\psi(\theta|x) + \mathbb{E}_{\tilde{q}_\phi(\theta|x)\tilde{p}(x)} \log\Big(1 - \mathrm{D}_\psi(\theta|x)\Big) \tag{4.11}$$

This ensures that the generator converges to the true posterior $p(\theta|x)$. Note that this approach follows a similar reasoning as in Che et al. (2020) [29] and Azadi et al. (2019) [7].

We demonstrate that GATSBI is competitive, but does not outperform state-of-the-art methods on benchmark tasks [102]. We also show that GATSBI returns well-calibrated posteriors on par with state-of-the-art SBI approaches on the shallow water model[8, 68]: a simulator that generates 100 (space)$\times$ 100 (time)-dimensional waves on the surface of a water body, given a 100-dimensional depth profile as parameter. We also show that GATSBI outperforms state-of-the-art methods on a 784-dimensional noisy camera model which produces blurred images given high-resolution images as inputs.

We briefly mention some work building on GATSBI: we tested GATSBI by applying it to a large-scale climate model of the El-Niño Southern Oscillations (ENSO) – the periodic warming and cooling of the central and eastern Pacific ocean. The model (called the Cane-Zebiak model) [190, 195] simulates tri-monthly ocean currents and surface winds over a spatial grid spanning the equatorial Pacific for a period of 20 years – the simulations are $\sim 2 \times 10^6$-dimensional. We use GATSBI to infer the initial conditions for the model (a 34-dimensional parameter space), a task that is out of reach for most other SBI methods due to the dimensionality of the observations. There are also some theoretical advances to GATSBI: Pacchiardi et al. (2022) [129] train a generative model of the posterior density with a scoring rule [49] instead of an adversarial loss. They show that the resulting networks perform comparably or better than GATSBI on the same tasks we describe above, but with far less computational overhead.

To summarize, in Ramesh et al. (2022) [147], we establish a connection between GANs, SBI and VAEs and clarify the similarities and differences of each of these machine learning tools. We also propose new GAN-based method for SBI: GATSBI, and show how to extend it to perform sequential inference. Finally, we demonstrate GATSBI's advantages over state-of-the-art SBI methods in high-dimensional parameter spaces. We thus open up new avenues of research for SBI with high-dimensional simulators, and enable new insights by clarifying its connections to VAEs and GANs.

Author contributions

This publication is co-authored with Jan-Matthis Lueckmann, Jan F. Boelts, Àlvaro Tejero-Cantero, David S. Greenberg, Pedro P. J. Gonçalves and Jakob H. Macke.

I developed the idea for the GAN approach and the sequential inference algorithm with help from all co-authors, and implemented it. Jakob H. Macke had the idea for the connection to variational inference and I implemented it. Jan-Matthis Lueckmann helped set up the benchmark

tasks and generate results for the other SBI algorithms, and produced all the results figures in the paper. Jan F. Boelts set up and executed the code for other SBI algorithms on the high-dimensional tasks and produced the results. Àlvaro Tejero-Cantero provided feedback and helped develop the mathematical proofs in this paper, and also made the illustration for GATSBI (Fig.1 of the paper). David S. Greenberg provided the idea and guided implementation of GATSBI for the shallow water model. Pedro P. J. Gonçalves and Jakob H. Macke provided feedback and guidance on developing the story for the paper, as well as exploring potential high-dimensional applications for GATSBI. All authors contributed to writing the paper.

4.4 Summary

We have discussed existing approaches for SBI and some of their drawbacks, including scalability to high-dimensional data, and the reliance on evaluable densities. We have also described GANs that have been applied to black-box simulator problems and how SBI is different from inference in a VAEs. In Ramesh et al. (2022) [147], we showed how the deficiencies in SBI approaches are filled in by our GAN approach, how it provides an insight into the connection between SBI and VAEs, and how this connection can be leveraged to set up a new sequential inference algorithm.

Chapter 5

The first rule of synaptic plasticity: there is no one rule

In this chapter, we provide background and describe the contributions in [146].

5.1 Background

An important challenge in neuroscience, is to understand the mechanisms underlying memory, learning and adaptation. Our brains are plastic i.e., connections between neurons in the brain are constantly updated [67, 139, 184], and this property is essential for survival in complex, ever-changing environments. Changes in neuronal connectivity alter the way neuronal networks process input stimuli (e.g. changes in the environment), and consequently, their output: this controls downstream behaviour in response to the input stimuli.

One of the important mechanisms by which changes in neuronal network changes are thought to be effected is synaptic plasticity. The junction between two neurons is known as a synapse. Incoming spikes pass from one neuron to the other via molecular signalling through the synapse. The synaptic strength or synaptic weight indicates the efficacy with which a signal is transferred from one neuron to the other via the synapse. However, the synaptic weight can undergo long-term or short-term modifications that are dependent on stimuli, brain state, experience, etc. In this way, synaptic plasticity occurs throughout the brain in various forms and under myriad conditions.

We here focus on theoretical explanations of synaptic plasticity i.e., we abstract the molecular mechanisms at the synapse into a single number representing the synaptic strength, and ask how this number changes as a function of the activity of the neurons that it connects to, inputs to these neurons, etc. Furthermore, theories for synaptic plasticity typically hypothesize a mathematical function describing how the synaptic strength changes as a function of the current synaptic strength, neuronal activity and other variables [33, 47, 182]. These mathematical functions are called synaptic update rules.

5.1.1 Theories for synaptic plasticity

Here, we briefly describe the prevalent theory for mechanisms of synaptic plasticity, based loosely on the review by Magee and Grienberger [107].

The most influential theory of plasticity postulates that synaptic plasticity is correlational: neurons that fire together cause the synapses that connect them to strengthen. This is called Hebbian plasticity [63]. This kind of plasticity was discovered to be in operation in the hippocampus [22,

93], where synchronous activation of two neurons (the pre- and post-synaptic neuron) selectively strengthened the synapse connecting them. Furthermore, there is some evidence that this kind of mechanism plays a role in the formation of associative memories [83, 111, 117, 121] and learning from sensory stimuli [27, 111, 175, 192]. Some update rules that implement Hebbian plasticity include changes in synaptic weights proportional to the correlation between the activity of pre- and post-synaptic neurons [69, 126, 153] and spike-timing dependent plasticity [17, STDP] i.e. changes in synaptic strength inversely proportional to the lag between pre- and post-synaptic spikes.

The rules described in the previous paragraph are all local: i.e. the update to synaptic weight depends exclusive on the activity of the neurons connected by the synapse. However, there is evidence that synaptic plasticity is modulated by more global factors e.g neurotransmitters such as dopamine [60, 85, 128], or via predictive coding [157]. Correspondingly, there exist update rule hypotheses that account for these effects e.g., Schultz et al. [158] describe plasticity as an error-correction mechanism for reward-seeking behaviour, and Sutton and Barto [168] use "internal" signals local to a synapse that are independent of the stimulus.

Neuromodulatory and correlational plasticity are mechanistic hypotheses for synaptic plasticity. However, it is also possible to theorize plasticity from a functional perspective, as a target-driven phenomenon: i.e., synaptic weights are changed with the goal of achieving a network state that produces a certain activity pattern or behaviour. Plasticity can be driven by a target that enforces specific activity patterns [167], encourages activity encoding stimulus information [155] or minimizes an activity-based error signal [154]. While the biological plausibility of supervised plasticity is a matter of debate [15], there is also some evidence that it occurs in networks of Purkenje cells [86] or locally within the electro-sensory lobe of electric fish [120].

We have described some hypothesized mechanisms of plasticity (correlation-based, neuromodulatory) as well as functional explanations of plasticity (to establish sparse activity, encode memory). While experimental studies provide qualitative support for theoretical mechanisms, and vice versa, fitting models of synaptic plasticity to data is a relatively new endeavour. In the following section, we discuss some approaches to learning the plasticity rule directly from data i.e. meta-learning plasticity rules.

5.1.2 Meta-learning plasticity rules

Theoretical formulations for mechanisms of plasticity are usually hand-crafted by interpreting experimental observations, but are not necessarily fit to experimental data. This approach provides only one of many possible explanations of the plasticity mechanism, it does not eliminate other hypotheses and thus, does not provide a robust or reliable understanding of synaptic plasticity.

As an alternative, recent approaches have attempted to replace hand-tuned rules with parametrized plasticity rules, and fit the parameters such that simulated network behaviour matches experimental data. We term this approach meta-learning: it allows for an algorithmic and systematic exploration of the space of potential rules that could fit the data, while optimizing for the parameters of the update rule.

For example, Lappalainen et al. [90] and Stepanyants et al. [166] algorithmically derive simplified analytical expressions based on network level quantities such as the firing rate, from detailed models of plasticity. Jordan et al. [77] define plasticity rules as combinations of symbolic expressions. They then use a genetic algorithm to find the correct combination of symbolic expressions that maximize a reward function defined based on a network level behaviour. Metz et al. [112], Tyulmankov et al. [179] and Lindsey et al. [95] all train their update rules to optimize an objective function designed for a network solving a task, but parametrize the update rule with a multi-layer

perceptron in the case of Metz et al. [112] or as a weighted combination of Hebbian rules in the latter two articles. Finally, Confavreux et al. [34] learn coefficients of a polynomial expansion or an MLP, while optimizing for network properties rather than task-solving behaviour.

The approaches we describe above all attempt to address the issue of finding robust mathematical formulations of the synaptic update rule directly from data. However, one major shortcoming of these approaches is that the parametrization of the rule and objective functions used to train it are derived based on the established theory in the field (subsection 5.1.1). Hence, they are subject to many of the same pitfalls of hand-tuning as the non-algorithmic approach. In the next section, we show how we can eliminate the need for hand-tuning by using GANs for meta-learning plasticity rules. Moreover, using the GAN-based approach, we demonstrate that space of plausible synaptic plasticity rules for observed data is sensitive both to the quality of the experimental data and the parametrization of the update rule. We thus show that the previous approaches hypothesizing singular update rules based on mechanisms or objective functions derived from interpretations of limited experimental data, could lead to incomplete or even erroneous insight into the neuronal system.

5.2 Publication: The first rule of synaptic plasticity: there is no one rule

This work is under review at the Thirty-sixth Conference on Neural Information Processing Systems 2022 and available as a preprint, at the time of submission of this thesis.

Understanding synaptic plasticity is essential for the study of mechanisms underlying learning and memory in the brain [1, 31]. However, measuring how synapses in the brain evolve with time, and how they influence the behaviour of neuronal networks is difficult. Thus the de-facto method for elucidating synaptic plasticity mechanisms and their role in shaping recordings from the brain has been theory. Specifically, this means using theoretical models of synaptic plasticity as hypotheses tested against observed data [33, 47, 54].

Theoretical models recast networks of neurons as the nodes of a graph, whose edges correspond to the synapses. The strength of a synapse connecting two neurons is indicated by a "weight". Synaptic mechanisms are captured by equations dictating how this weight changes with time, and are termed plasticity rules. These equations are usually formulated from experimental observations and biological plausibility arguments. However, with increasingly rich experimental data available from large-scale recordings of populations of neurons across several brain regions, this approach falls short of explaining all the observed variability in the data. Crucially, this theoretical approach relies on a subjective interpretation of the data followed by hand-tuning for formulating hypotheses. Furthermore, it provides only one possible explanation of the observed data – these models typically involve a single update rule that generates desired network behaviour [96, 196]. With this approach, there is no guarantee that if there were more data available, or if we perturbed the observed quantities, that these hypothesized rules would still produce valid network behaviour. In other words, these approaches provide nescessary but not sufficient explanations for synaptic plasticity, which may not be robust.

A more reasonable approach would be to numerically explore the space of all possible rules that could underlie observed data – i.e. "meta-learn" plasticity rules directly from the data, by parametrizing the plasticity rules and fitting these parameters to data. With this approach, we may take advantage of the wide range of data available as well as improvements in numerical approaches, and leverage them effectively to study synaptic plasticity. The success of this approach

requires that we have both a sufficiently flexible parametrization of the plasticity rules, as well as an objective function (or loss function) capable of capturing the relevant information in the data for meta-learning the rules.

There have been several approaches that focus on flexible rule parametrization for capturing network behaviour [34, 77, 95, 112, 179]: all of which use hand-tuned loss functions derived from hypotheses about the neuronal network to optimize the rule parameters. While this approach is successful, it moves the problems of sufficiency and robustness from the formulation of the plasticity rule in previous theoretical approaches, to the formulation of the loss function.

In Ramesh et. al 2022 [146], we propose to meta-learn both the update rule and the loss function directly from data, by recasting the problem in the GANs framework. Specifically, we use deep neural networks (DNNs) to parametrize the update rule as well as the loss function, and train the two networks using an adversarial objective function. This simultaneously eliminates the need to hand-craft either the update rule or the loss function, and instead enables us to meta-learn them by comparing simulated data to observed data.

The GAN frameworks works as follows: we simulate data from a biological rate network of N presynaptic neurons with activity $x_j, j = 1 \dots N$, M postsynaptic neurons with activity $y_i, i = 1 \dots M$, and synaptic weights ω_{ij}. The postsynaptic activity at (discretized) time t is updated as follows:

$$y_i^t = \sum_j \omega_{ij}^t x_j^t \tag{5.1}$$

The weights are updated using a plasticity rule parametrized by ϕ:

$$\Delta\omega_{ij}^t = h_\phi(x_j^t, y_i^t, \omega_{ij}^t) \tag{5.2}$$

The rate network and the parametrized rule combined form the GAN generator, and a forward pass through it produces postsynaptic activity trace for T timesteps $\mathbf{y} = \{x_j^t\}_{i=1,t=1}^{N,T}$, obtained by alternatively updating the postsynaptic activity and synaptic weights using Equation 5.1 and Equation 5.2.

We generate our target data $\mathbf{y}'$ in the same manner, as for the generator, but replace the parametrized update rule with a known rule i.e., Oja's rule [126]:

$$\Delta\omega_{ij}^t = x_j^t * y_i^t - (y_i^t)^2 \omega_{ij}^t \tag{5.3}$$

This is a Hebbian learning rule, with well-characterized effects on the postsynaptic activity: when used to update the synaptic weights ω_{ij} of the feedforward rate network in Equation 5.1, it causes ω_{ij} to converge to the first principal component PC_1 of the presynaptic activity $\mathbf{x} = \{x_i\}_{i=1}^N$ [126]. Concurrently, under Oja's rule, the postsynaptic activity $\mathbf{y}$ converges to a projection of $\mathbf{x}$ onto its first principal component.

In lieu of an objective function, we define a discriminator D_ψ which takes as input $\mathbf{y}$ to classify as fake, or $\mathbf{y}'$ to classify as real. We train the parameters ϕ and ψ using the minimax loss:

$$\phi^*, \psi^* = \min_\phi \max_\psi \mathbb{E}_{p(\mathbf{y}')} \log \mathrm{D}_\psi(\mathbf{y}') + \mathbb{E}_{p_\phi(\mathbf{y})} \log\big(1 - \mathrm{D}_\psi(\mathbf{y})\big). \tag{5.4}$$

When the two networks are trained with this loss, they parameters ϕ would theoretically converge to a parametrization of the rule that produces data from the biological rate network with the same statistics as the data from Oja's rule. Thus, we can meta-learn the parameters ϕ without having to hand-tune the data features or the objective function relevant to the underlying groundtruth rule.

We then hypothesize that multiple rules could generate the similar postsynaptic activity traces as Oja's rule. We show how it was possible to explore this space of possible rules using different parametrizations of the update rule:

- *Local MLP*: the plasticity rule is parametrized by a 3-layer MLP, i.e., $h_\phi(\cdot) = \mathrm{MLP}_\phi(\omega_{ij}^t, x_j^t, y_i^t)$. This MLP represents a *local update*, i.e., it transforms each x_j^t, y_i^t and ω_{ij}^t in the same way, independently of the indices i, j and t.
- *Oja + local MLP*: $h_\phi(\cdot) = \mathrm{MLP}_\phi(\omega_{ij}^t, x_j^t, y_i^t) + x_j^t * y_i^t - (y_i^t)^2 \omega_{ij}^t$. This learning rule is "biased" since, by construction, it is initialised close to the ground-truth solution and any non-zero outputs of the MLP are perturbations to Oja's rule.
- *Semi-global MLP* computes the synaptic weight update $\Delta\omega_{ij}^t$ for a single synapse. It takes into account the mean presynaptic activity and the mean across the network synaptic weights at the current time step, in addition to the local presynaptic activity x_j^t, synaptic weight ω_{ij}^t and postsynaptic activity y_i^t: $\Delta\omega_{ij}^t = h_\phi(\cdot) = \mathrm{MLP}_\phi(\omega_{ij}^t, x_j^t, y_i^t, \frac{1}{N}\sum_j x_j^t, \frac{1}{N}\sum_j \omega_{ij})$.
- *Global MLP* takes into account all pre- and postsynaptic activities and synaptic weights: $\Delta\omega_{ij} = h_\phi(\cdot) = \mathrm{MLP}_\phi(\{\omega_{ij}^t, x_j^t, y_i^t\}_{i=1,j=1}^{M,N})$.

We also show how different perturbations to the training data for the network contribute to differences in the learned rules. For this, we generated groundtruth data with Oja's rule, but made changes to the linear network in Equation 5.1:

- We used $M = 3$ pre-synaptic neurons in the rate netowrk, and $N = 1$ postsynaptic neuron
- We used $M = 39$ pre-synaptic neurons and $N = 1$ postsynaptic neuron
- We used $M = 3$ pre-synaptic neurons in the rate network, and $N = 1$ postsynaptic neuron, but introduced noise in the postsynaptic activity: $y_i^t = \sum_j \omega_{ij}^t x_j^t + \epsilon, \epsilon \sim \mathcal{N}(0, 0.25^2)$

We found that each of these differences in rule parametrization and training data led to different learned rules, even though all rules produced postsynaptic activity traces with same statistics as the groundtruth data with Oja's rule. More importantly, we find that each of these perturbations, particularly those to the training data result in rules that generalise to test data that is different from the training data. This indicates that the learned rules do capture the dynamics that are preserved across different datasets and parametrizations (namely, the same dynamics that Oja's rule produces). However, the plethora of rules we find also indicate that even in this simplified system, Oja's rule is not the only plasticity rule capable of producing the same activity dynamics – in other words, the space of plasticity rules, at least in this setting, are degenerate and underconstrained by postsynaptic activity traces alone.

Our results signal the need for a shift from thinking about plasticity mechanisms as individual equations explaining observed data, to families or manifolds of rules producing the same activity at the network level. Furthermore, this manifold may be influenced by our parametrization of the functions, or the quality of the data on which they are trained: this indicates the requirement of estimating uncertainty over these meta-learned rules, as well as characterizing properties conserved across different learned rules. The GAN framework we propose has several advantages, beyond the flexibility we outlined in previous paragraphs: first, the GANs capture the statistics of the observed data, rather than matching individual generated and simulated traces. This could ensure that the rules are not overfit to anomalies in the observed data due to changing attentional state of the

animal or noise in the recording equipment (as indicated by our experiments with perturbing the training data). Thus, in addition to the flexibility it confers, the GANs may also lead to rules that are more robust to idiosyncrasies in data. Second, the GANs allow for conditioning variables, thus allowing for learned rules to be dependent on context or experimental conditions under which the data was recorded. Finally, the GAN approach for meta-learning plasticity rules shares many similarities with neural ordinary differential equations [30], physics-informed GANs [191] and unsupervised reinforcement learning [57]. Thus, we might be able to leverage techniques from all these applications in order to improve our GANs framework.

Author contributions

This publication is co-authored with Basile Confavreux, Pedro P. J. Gonçalves, Tim P. Vogels and Jakob H. Macke.

I developed the idea for the GAN approach and its application to Oja's rule with help from all co-authors, and implemented it. Basile Confavreux provided the idea for the model parametrization experiments, implemented the biological rate network and some of the post-hoc analysis on the learned rules, made Figure 1 from the paper and provided constant feedback on the GAN implementation. Pedro P. J. Gonçalves provided both low-level and high level feedback on the method, results and post-training analysis on the learned rules. Pedro P. J. Gonçalves, Jakob H. Macke and Tim P. Vogels provided feedback and guidance on the model parametrization experiments and guidance in developing the story for the paper. All authors contributed to writing the paper.

Chapter 6

Conclusion

Generative adversarial networks are generally thought of as machine learning tools for generating realistic data. They are capable of reproducing a wide variety of data types for many different scientific applications. The flexibility and scalability of GANs stem both from the fact that their training relies on on samples from the target dataset, and that there are no theoretical limitations on the architecture of the networks in order for them to work in principle.

The most well-known GANs are generators of images [74, 79, 198], medical images [12, 187, 199], translators of text to audio [39], etc. However, in this thesis, we have explored their use as estimators of arbitrarily complex densities – i.e., any target distribution that is structured and high-dimensional. We have shown that GANs are useful tools for reproducing "realistic" data in domains beyond image generation: in neuroscience and climate science. However, this flexibility enables far more scientific insight than merely generating realistic data: it enables us to study the underlying systems that generate this data in new ways.

For example, in the application for reproducing neural population activity, we showed that a generative model that produces more realistic spike trains provides a different explanation of the stimulus features that drive this activity, compared to models that only reproduce specific statistics of the spike train. With GANs for simulation-based inference (SBI), we showed that in addition to extending the scope of SBI to high-dimensional parameter spaces, GANs also reveal a connection between SBI and variational auto-encoders (VAEs) that can be leveraged for new avenues of sequential inference and SBI applications. With GANs for inferring plasticity mechanisms, we showed how changes to data and model parametrization influence our understanding of plasticity.

However, training GANs to perform these tasks is notoriously hard – they are sensitive to initial conditions, prone to mode collapse, and require an extensive amount of hyperparameter tuning to converge. Moreover, they require a lot of computational expense and capacity, that scales with the dimensionality of the data. While they are immensely flexible and the extent of their utility is yet to be fully plumbed, it is worth asking whether the computational cost of using them is commensurate with the insights and flexibility they provide. These problems also limit reproducibility and the widespread use of computational approaches developed using GANs.

There are several lines of research to elucidate how GANs work, encompassing their convergence properties [43, 164], how to reason about their behaviour under different data regimes[194] and to make training them easier in practice [72, 156]. The combined results of these endeavours could result in more usable GANs. There might also emerge other architectures such as denoising diffusion models [65] or transformers [181], that are capable of the same kinds of flexibility, without the associated computational or tuning costs.

In conclusion, it is clear that there is a lot of opportunity for research and applications of GANs

that have yet to be discovered. This is a machine learning tool that can be leveraged for a variety of problems and applications, that are quite different from the traditional computer vision or natural language processing frameworks that they are renowned for – some of which we have elucidated in this thesis. When used carefully, and in cases where the computational costs are offset by the degree of scientific progress and insight they enable, GANs can be enormously useful.

Bibliography

[1] Larry F Abbott and Sacha B Nelson. "Synaptic plasticity: taming the beast". In: *Nature neuroscience* 3.11 (2000), pp. 1178–1183.

[2] Jonas Adler and Ozan Öktem. *Deep Bayesian Inversion*. 2018. arXiv: 1811.05910 [stat.ML].

[3] Edgar D Adrian and Yngve Zotterman. "The impulses produced by sensory nerve endings: Part 3. Impulses set up by Touch and Pressure". In: *The Journal of physiology* 61.4 (1926), p. 465.

[4] Amirmasoud Ahmadi, Mahsa Behroozi, Vahid Shalchyan, and Mohammad Reza Daliri. "Rat navigation by stimulating somatosensory cortex". In: *Journal of Bionic Engineering* 16.5 (2019), pp. 931–942.

[5] Takafumi Arakaki, G. Barello, and Yashar Ahmadian. "Inferring neural circuit structure from datasets of heterogeneous tuning curves". In: *PLoS Computational Biology* 15.4 (2019).

[6] Martin Arjovsky, Soumith Chintala, and Léon Bottou. "Wasserstein Generative Adversarial Networks". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. PMLR, Aug. 2017, pp. 214–223.

[7] Samaneh Azadi, Catherine Olsson, Trevor Darrell, Ian Goodfellow, and Augustus Odena. "Discriminator Rejection Sampling". In: *arXiv:1810.06758 [cs, stat]* (2019). arXiv: 1810.06758.

[8] Jan O Backhaus. "A semi-implicit scheme for the shallow water equations for application to shelf sea modelling". In: *Continental Shelf Research* 2.4 (1983), pp. 243–254.

[9] Mohammad Bashiri, Edgar Walker, Konstantin-Klemens Lurz, Akshay Jagadish, Taliah Muhammad, Zhiwei Ding, Zhuokun Ding, Andreas Tolias, and Fabian Sinz. "A flow-based latent state generative model of neural population responses to natural images". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 15801–15815.

[10] Demian Battaglia, Annette Witt, Fred Wolf, and Theo Geisel. "Dynamic effective connectivity of inter-areal brain circuits". In: *PLoS computational biology* 8.3 (2012), e1002438.

[11] Eleanor Batty, Josh Merel, Nora Brackbill, Alexander Heitman, Alexander Sher, Alan Litke, EJ Chichilnisky, and Liam Paninski. "Multilayer recurrent network models of primate retinal ganglion cell responses". In: (2016).

[12] Christoph Baur, Shadi Albarqouni, and Nassir Navab. *MelanoGANs: High Resolution Skin Lesion Synthesis with GANs*. 2018. DOI: 10.48550/ARXIV.1804.04338.

[13] Mark A Beaumont, Wenyang Zhang, and David J Balding. "Approximate Bayesian computation in population genetics". In: *Genetics* 162.4 (2002), pp. 2025–2035.

[14] Paul Beier and Dale R McCullough. "Motion-sensitive radio collars for estimating white-tailed deer activity". In: *The Journal of Wildlife Management* (1988), pp. 11–13.

[15] Yoshua Bengio, Dong-Hyun Lee, Jorg Bornschein, Thomas Mesnard, and Zhouhan Lin. "Towards biologically plausible deep learning". In: *arXiv preprint arXiv:1502.04156* (2015).

[16] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. "Estimating or propagating gradients through stochastic neurons for conditional computation". In: *arXiv preprint arXiv:1308.3432* (2013).

[17] Guo-qiang Bi and Mu-ming Poo. "Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type". In: *Journal of neuroscience* 18.24 (1998), pp. 10464–10472.

[18] NG Bibikov. "" Novelty" neurons in the frog auditory system". In: *Zhurnal Vysshei Nervnoi Deiatelnosti Imeni IP Pavlova* 27.5 (1977), pp. 1075–1082.

[19] C M Bishop. "Mixture density networks". In: *Technical Report. Aston University, Birmingham* (1994).

[20] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*. Vol. 4. 4. Springer, 2006.

[21] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. "Variational inference: A review for statisticians". In: *Journal of the American statistical Association* 112.518 (2017), pp. 859–877.

[22] Tim VP Bliss and Terje Lømo. "Long-lasting potentiation of synaptic transmission in the dentate area of the anaesthetized rabbit following stimulation of the perforant path". In: *The Journal of physiology* 232.2 (1973), pp. 331–356.

[23] Michael GB Blum and Olivier François. "Non-linear regression models for Approximate Bayesian Computation". In: *Statistics and Computing* 20.1 (2010), pp. 63–73.

[24] Oliver J Britton, Alfonso Bueno-Orovio, Karel Van Ammel, Hua Rong Lu, Rob Towart, David J Gallacher, and Blanca Rodriguez. "Experimentally calibrated population of models predicts and explains intersubject variability in cardiac cellular electrophysiology". In: *Proceedings of the National Academy of Sciences* 110.23 (2013), E2098–E2105.

[25] Andrew Brock, Jeff Donahue, and Karen Simonyan. "Large scale GAN training for high fidelity natural image synthesis". In: *arXiv preprint arXiv:1809.11096* (2018).

[26] Daniel Bush and Neil Burgess. "Neural oscillations: phase coding in the absence of rhythmicity". In: *Current Biology* 29.2 (2019), R55–R57.

[27] Luis Carrillo-Reid, Weijian Yang, Yuki Bando, Darcy S Peterka, and Rafael Yuste. "Imprinting and recalling cortical ensembles". In: *Science* 353.6300 (2016), pp. 691–694.

[28] Sangwon Chae, Sungjun Kwon, and Donghyun Lee. "Predicting infectious disease using deep learning and big data". In: *International journal of environmental research and public health* 15.8 (2018), p. 1596.

[29] Tong Che, Ruixiang Zhang, Jascha Sohl-Dickstein, Hugo Larochelle, Liam Paull, Yuan Cao, and Yoshua Bengio. "Your GAN is Secretly an Energy-based Model and You Should use Discriminator Driven Latent Sampling". In: *arXiv:2003.06060 [cs, stat]* (2020).

[30] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. *Neural Ordinary Differential Equations*. 2018. DOI: 10.48550/ARXIV.1806.07366.

[31] Ami Citri and Robert C. Malenka. "Synaptic Plasticity: Multiple Forms, Functions, and Mechanisms". In: *Neuropsychopharmacology* 33 (2008), pp. 18–41.

[32] Andy Clark. “Whatever next? Predictive brains, situated agents, and the future of cognitive science”. In: *Behavioral and brain sciences* 36.3 (2013), pp. 181–204.

[33] Claudia Clopath, Lars Büsing, Eleni Vasilaki, and Wulfram Gerstner. “Connectivity reflects coding: a model of voltage-based STDP with homeostasis”. In: *Nature Neuroscience* 13 (2010), pp. 344–352.

[34] Basile Confavreux, Friedemann Zenke, Everton J Agnes, Timothy Lillicrap, and Tim P Vogels. “A meta-learning approach to (re) discover plasticity rules that carve a desired function into a neural network”. In: *Advances in Neural Information Processing Systems 34 (NeurIPS)* (2020).

[35] Kyle Cranmer, Johann Brehmer, and Gilles Louppe. “The frontier of simulation-based inference”. In: *Proceedings of the National Academy of Sciences* 117.48 (2020), pp. 30055–30062.

[36] Arthur P Dempster, Nan M Laird, and Donald B Rubin. “Maximum likelihood from incomplete data via the EM algorithm”. In: *Journal of the Royal Statistical Society: Series B (Methodological)* 39.1 (1977), pp. 1–22.

[37] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. “Imagenet: A large-scale hierarchical image database”. In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[38] Petar M Djuric, Jayesh H Kotecha, Jianqui Zhang, Yufei Huang, Tadesse Ghirmai, Mónica F Bugallo, and Joaquin Miguez. “Particle filtering”. In: *IEEE signal processing magazine* 20.5 (2003), pp. 19–38.

[39] Chris Donahue, Julian McAuley, and Miller Puckette. “Adversarial audio synthesis”. In: *arXiv preprint arXiv:1802.04208* (2018).

[40] Jeff Donahue, Trevor Darrell, and Philipp Krähenbühl. “Adversarial feature learning”. In: *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings* (2019), pp. 1–18.

[41] Conor Durkan, George Papamakarios, and Iain Murray. “Sequential Neural Methods for Likelihood-free Inference”. In: *Bayesian Deep Learning Workshop at Neural Information Processing Systems* (2018). eprint: 1811.08723.

[42] Jesse Engel, Kumar Krishna Agrawal, Shuo Chen, Ishaan Gulrajani, Chris Donahue, and Adam Roberts. “GANSynth: Adversarial Neural Audio Synthesis”. In: *International Conference on Learning Representations*. 2019.

[43] William Fedus, Mihaela Rosca, Balaji Lakshminarayanan, Andrew M Dai, Shakir Mohamed, and Ian Goodfellow. “Many paths to equilibrium: GANs do not need to decrease a divergence at every step”. In: *arXiv preprint arXiv:1710.08446* (2017).

[44] Karl Friston. “Learning and inference in the brain”. In: *Neural Networks* 16.9 (2003), pp. 1325–1352.

[45] Raghavendra Gadagkar. “Dominance Hierarchy and Division of Labour in the Social Wasp Ropalidia marginata (Lep.)(Hymenoptera: Vespidae)”. In: *Current Science* (1980), pp. 772–775.

[46] Paul A Gagniuc. *Markov chains: from theory to implementation and experimentation*. John Wiley & Sons, 2017.

[47] Wulfram Gerstner, Richard Kempter, J. Leo van Hemmen, and Hermann Wagner. “A neuronal learning rule for sub-millisecond temporal coding”. In: *Nature* 383 (1996), pp. 76–78.

[48] Ivan Alexander Getting. “Perspective/navigation-the global positioning system”. In: *IEEE spectrum* 30.12 (1993), pp. 36–38.

[49] Tilmann Gneiting and Adrian E Raftery. “Strictly proper scoring rules, prediction, and estimation”. In: *Journal of the American statistical Association* 102.477 (2007), pp. 359–378.

[50] 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 27*. Ed. by Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger. Curran Associates, Inc., 2014, pp. 2672–2680.

[51] Cheryl L Grady, John W Van Meter, Jose Ma Maisog, Pietro Pietrini, Jack Krasuski, and Josef P Rauschecker. “Attention-related modulation of activity in primary and secondary auditory cortex”. In: *Neuroreport* 8.11 (1997), pp. 2511–2516.

[52] Einat Granot-Atedgi, Gašper Tkačik, Ronen Segev, and Elad Schneidman. “Stimulus-dependent maximum entropy models of neural population codes”. In: *PLoS computational biology* 9.3 (2013), e1002922.

[53] Will Grathwohl, Dami Choi, Yuhuai Wu, Geoffrey Roeder, and David Duvenaud. “Backpropagation through the Void: Optimizing control variates for black-box gradient estimation”. In: (2017). arXiv: 1711.00123.

[54] Michael Graupner and Nicolas Brunel. “Calcium-based plasticity model explains sensitivity of synaptic changes to spike pattern, rate, and dendritic location”. In: *Proceedings of the National Academy of Sciences* 109.10 (2012), pp. 3991–3996.

[55] David Greenberg, Marcel Nonnenmacher, and Jakob Macke. “Automatic Posterior Transformation for Likelihood-Free Inference”. In: *Proceedings of the 36th International Conference on Machine Learning*. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 2404–2414.

[56] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. “Improved training of wasserstein gans”. In: *Advances in neural information processing systems* 30 (2017).

[57] Abhishek Gupta, Benjamin Eysenbach, Chelsea Finn, and Sergey Levine. *Unsupervised Meta-Learning for Reinforcement Learning*. 2018. DOI: 10.48550/ARXIV.1806.04640.

[58] Michael U. Gutmann and Jukka Corander. *Bayesian Optimization for Likelihood-Free Inference of Simulator-Based Statistical Models*. 2015. arXiv: 1501.03291 [stat.ML].

[59] Juergen Haag and Alexander Borst. “Encoding of visual motion information and reliability in spiking and graded potential neurons”. In: *Journal of Neuroscience* 17.12 (1997), pp. 4809–4819.

[60] Trevor J Hamilton, B Matthew Wheatley, D Barry Sinclair, Madeline Bachmann, Matthew E Larkum, and William F Colmers. “Dopamine modulates synaptic plasticity in dendrites of rat and human dentate granule cells”. In: *Proceedings of the National Academy of Sciences* 107.42 (2010), pp. 18185–18190.

[61] Harry H Harman. *Modern factor analysis*. University of Chicago press, 1976.

[62] Gregory M Harry, LIGO Scientific Collaboration, et al. "Advanced LIGO: the next generation of gravitational wave detectors". In: *Classical and Quantum Gravity* 27.8 (2010), p. 084006.

[63] Donald Olding Hebb. *The organization of behavior: A neuropsychological theory*. Psychology Press, 2005.

[64] Joeri Hermans, Volodimir Begy, and Gilles Louppe. "Likelihood-free MCMC with Approximate Likelihood Ratios". In: *Proceedings of the 37th International Conference on Machine Learning*. Vol. 98. Proceedings of Machine Learning Research. PMLR, 2020.

[65] Jonathan Ho, Ajay Jain, and Pieter Abbeel. "Denoising diffusion probabilistic models". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 6840–6851.

[66] Alan L Hodgkin and Andrew F Huxley. "A quantitative description of membrane current and its application to conduction and excitation in nerve". In: *The Journal of physiology* 117.4 (1952), p. 500.

[67] Christian Hölscher. "Synaptic plasticity and learning and memory: LTP and beyond". In: *Journal of neuroscience research* 58.1 (1999), pp. 62–75.

[68] James R Holton. "An introduction to dynamic meteorology". In: *American Journal of Physics* 41.5 (1973), pp. 752–754.

[69] John J Hopfield. "Neural networks and physical systems with emergent collective computational abilities." In: *Proceedings of the national academy of sciences* 79.8 (1982), pp. 2554–2558.

[70] Joseph B Hopfinger, Michael H Buonocore, and George R Mangun. "The neural mechanisms of top-down attentional control". In: *Nature neuroscience* 3.3 (2000), pp. 284–291.

[71] Andrew G. Howard. *Some Improvements on Deep Convolutional Neural Network Based Image Classification*. 2013. DOI: 10.48550/ARXIV.1312.5402.

[72] Ferenc Huszár. *How (not) to Train your Generative Model: Scheduled Sampling, Likelihood, Adversary?* 2015. DOI: 10.48550/ARXIV.1511.05101.

[73] Aapo Hyvärinen. "Independent component analysis: recent advances". In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 371.1984 (2013), p. 20110534.

[74] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. "Image-to-Image Translation with Conditional Adversarial Networks". In: *arXiv preprint arXiv:1611.07004* (2016).

[75] Rafael Izbicki, Ann Lee, and Chad Schafer. "High-dimensional density ratio estimation with extensions to approximate likelihood computation". In: *Artificial Intelligence and Statistics*. 2014, pp. 420–429.

[76] Vinay Jethava and Devdatt Dubhashi. "Easy High-Dimensional Likelihood-Free Inference". In: *arXiv preprint arXiv:1711.11139* (2017).

[77] Jakob Jordan, Maximilian Schmidt, Walter Senn, and Mihai A Petrovici. "Evolving interpretable plasticity for spiking networks". In: *eLife* 10:e66273 (2021).

[78] James T Kadonaga. "The transformation of the DNA template in RNA polymerase II transcription: a historical perspective". In: *Nature structural & molecular biology* 26.9 (2019), pp. 766–770.

[79] Tero Karras, Samuli Laine, and Timo Aila. “A Style-Based Generator Architecture for Generative Adversarial Networks”. In: *arXiv:1812.04948* (2018).

[80] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. “Analyzing and improving the image quality of stylegan”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 8110–8119.

[81] Christoph Kayser, Marcelo A Montemurro, Nikos K Logothetis, and Stefano Panzeri. “Spike-phase coding boosts and stabilizes information carried by spatial and temporal spike patterns”. In: *Neuron* 61.4 (2009), pp. 597–608.

[82] Dongjun Kim, Weonyoung Joo, Seungjae Shin, and Il-Chul Moon. “Adversarial Likelihood-Free Inference on Black-Box Generator”. In: *arXiv preprint arXiv:2004.05803* (2020).

[83] Woong Bin Kim and Jun-Hyeong Cho. “Encoding of discriminative fear memory by input-specific LTP in the amygdala”. In: *Neuron* 95.5 (2017), pp. 1129–1146.

[84] Diederik P Kingma and Max Welling. “Auto-encoding variational bayes”. In: *arXiv preprint arXiv:1312.6114* (2013).

[85] Giacomo Koch, Zaira Esposito, Claudia Codecà, Francesco Mori, Hajime Kusayanagi, Fabrizia Monteleone, Francesco Di Lorenzo, Giorgio Bernardi, and Alessandro Martorana. “Altered dopamine modulation of LTD-like plasticity in Alzheimer’s disease patients”. In: *Clinical neurophysiology* 122.4 (2011), pp. 703–707.

[86] Dimitar Kostadinov, Maxime Beau, Marta Blanco-Pozo, and Michael Häusser. “Predictive and reactive reward signals conveyed by climbing fiber inputs to cerebellar Purkinje cells”. In: *Nature neuroscience* 22.6 (2019), pp. 950–962.

[87] Athanasios Kousathanas, Christoph Leuenberger, Jonas Helfer, Mathieu Quinodoz, Matthieu Foll, and Daniel Wegmann. *Likelihood-free inference in high-dimensional models*. 2015. arXiv: 1507.08612 [stat.ME].

[88] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger. Vol. 25. Curran Associates, Inc., 2012.

[89] Matt J Kusner and José Miguel Hernández-Lobato. “Gans for sequences of discrete elements with the gumbel-softmax distribution”. In: *arXiv preprint arXiv:1611.04051* (2016).

[90] Janne Lappalainen, Juliane Herpich, and Christian Tetzlaff. “A theoretical framework to derive simple, Firing-Rate-Dependent mathematical models of synaptic plasticity”. In: *Frontiers in computational neuroscience* 13 (2019), p. 26.

[91] Brodie AJ 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”. In: *Science Advances* 4.1 (2018), e1701676.

[92] Christiane Lemieux. “Control variates”. In: *Wiley StatsRef: Statistics Reference Online* (2014), pp. 1–8.

[93] WB Levy and O Steward. “Temporal contiguity requirements for long-term associative potentiation/depression in the hippocampus”. In: *Neuroscience* 8.4 (1983), pp. 791–797.

[94] Chun-Liang Li, Wei-Cheng Chang, Yu Cheng, Yiming Yang, and Barnabás Póczos. "Mmd gan: Towards deeper understanding of moment matching network". In: *Advances in neural information processing systems* 30 (2017).

[95] Jack Lindsey and Ashok Litwin-Kumar. "Learning to Learn with Feedback and Local Plasticity". In: *Advances in Neural Information Processing Systems 34 (NeurIPS)* (2020).

[96] Ashok Litwin-Kumar and Brent Doiron. "Formation and maintenance of neuronal assemblies through synaptic plasticity". In: *Nature Communications* 5 (2014).

[97] Jun S Liu, Rong Chen, and Tanya Logvinenko. "A theoretical framework for sequential importance sampling with resampling". In: *Sequential Monte Carlo methods in practice*. Springer, 2001, pp. 225–246.

[98] Gilles Louppe, Joeri Hermans, and Kyle Cranmer. "Adversarial variational optimization of non-differentiable simulators". In: *The 22nd International Conference on Artificial Intelligence and Statistics*. 2017, pp. 1438–1447.

[99] Mario Lucic, Karol Kurach, Marcin Michalski, Sylvain Gelly, and Olivier Bousquet. "Are gans created equal? a large-scale study". In: *Advances in neural information processing systems* 31 (2018).

[100] Steven J Luck, Leonardo Chelazzi, Steven A Hillyard, and Robert Desimone. "Neural mechanisms of spatial selective attention in areas V1, V2, and V4 of macaque visual cortex". In: *Journal of neurophysiology* 77.1 (1997), pp. 24–42.

[101] 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*. Vol. 96. Proceedings of Machine Learning Research. PMLR, 2019, pp. 32–53.

[102] Jan-Matthis Lueckmann, Jan Boelts, David Greenberg, Pedro Goncalves, and Jakob Macke. "Benchmarking Simulation-Based Inference". In: *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*. Ed. by Arindam Banerjee and Kenji Fukumizu. Vol. 130. Proceedings of Machine Learning Research. PMLR, Apr. 2021, pp. 343–351.

[103] 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 30*. Curran Associates, Inc., 2017, pp. 1289–1299.

[104] Jakob H Macke, Philipp Berens, Alexander S Ecker, Andreas S Tolias, and Matthias Bethge. "Generating Spike Trains with Specified Correlation Coefficients". In: *Neural Computation* 423 (2009), pp. 397–423.

[105] Jakob H Macke, Lars Buesing, John P Cunningham, Byron M Yu, Krishna V Shenoy, and Maneesh Sahani. "Empirical models of spiking in neural populations". In: *Advances in neural information processing systems* 24 (2011).

[106] Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. "The concrete distribution: a continuous relaxation of discrete random variables". In: *ArXiv* (2016). arXiv: arXiv:1611.00712v3.

[107] Jeffrey C. Magee and Christine Grienberger. "Synaptic Plasticity Forms and Functions". In: *Annual Review of Neuroscience* 43.1 (2020), pp. 95–117.

[108] Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. "Least squares generative adversarial networks". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2794–2802.

[109] Paul Marjoram and Simon Tavaré. "Modern computational approaches for analysing molecular genetic variation data". In: *Nature Reviews Genetics* 7.10 (2006), pp. 759–770.

[110] Lane McIntosh, Niru Maheswaranathan, Aran Nayebi, Surya Ganguli, and Stephen Baccus. "Deep Learning Models of the Retinal Response to Natural Scenes". In: *Advances in neural information processing systems* 29 (Feb. 2017).

[111] MG McKernan and P Shinnick-Gallagher. "Fear conditioning induces a lasting potentiation of synaptic currents in vitro". In: *Nature* 390.6660 (1997), pp. 607–611.

[112] Luke Metz, Niru Maheswaranathan, Brian Cheung, and Jascha Sohl-Dickstein. "Learning unsupervised learning rules". In: *arXiv* (2018).

[113] Mehdi Mirza and Simon Osindero. "Conditional generative adversarial nets". In: *arXiv preprint arXiv:1411.1784* (2014).

[114] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. "Spectral Normalization for Generative Adversarial Networks". In: *CoRR* abs/1802.05957 (2018).

[115] Manuel Molano-Mazon, Arno Onken, Eugenio Piasini, and Stefano Panzeri. "Synthesizing realistic neural population activity patterns using Generative Adversarial Networks". In: *ICLR* (2018).

[116] Marcelo A Montemurro, Malte J Rasch, Yusuke Murayama, Nikos K Logothetis, and Stefano Panzeri. "Phase-of-firing coding of natural visual stimuli in primary visual cortex". In: *Current biology* 18.5 (2008), pp. 375–380.

[117] RGM Morris, Elizabeth Anderson, GS a Lynch, and Michel Baudry. "Selective impairment of learning and blockade of long-term potentiation by an N-methyl-D-aspartate receptor antagonist, AP5". In: *Nature* 319.6056 (1986), pp. 774–776.

[118] Youssef Mroueh and Tom Sercu. "Fisher gan". In: *Advances in Neural Information Processing Systems* 30 (2017).

[119] James R Muller, Andrew B Metha, John Krauskopf, and Peter Lennie. "Rapid adaptation in visual cortex to the structure of images". In: *Science* 285.5432 (1999), pp. 1405–1408.

[120] Salomon Z Muller, Abigail N Zadina, LF Abbott, and Nathaniel B Sawtell. "Continual learning in a multi-layer network of an electric fish". In: *Cell* 179.6 (2019), pp. 1382–1392.

[121] Kazu Nakazawa, Thomas J McHugh, Matthew A Wilson, and Susumu Tonegawa. "NMDA receptors, place cells and hippocampal spatial memory". In: *Nature Reviews Neuroscience* 5.5 (2004), pp. 361–372.

[122] John F Nash Jr. "Equilibrium points in n-person games". In: *Proceedings of the national academy of sciences* 36.1 (1950), pp. 48–49.

[123] Emre O Neftci, Hesham Mostafa, and Friedemann Zenke. "Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks". In: *IEEE Signal Processing Magazine* 36.6 (2019), pp. 51–63.

[124] John Ashworth Nelder and Robert WM Wedderburn. "Generalized linear models". In: *Journal of the Royal Statistical Society: Series A (General)* 135.3 (1972), pp. 370–384.

[125] Augustus Odena, Christopher Olah, and Jonathon Shlens. *Conditional Image Synthesis With Auxiliary Classifier GANs*. 2016. DOI: 10.48550/ARXIV.1610.09585.

[126] Erkki Oja. "Simplified neuron model as a principal component analyzer". In: *Journal of mathematical biology* 15.3 (1982), pp. 267–273.

[127] Victor M.-H. Ong, David J. Nott, Minh-Ngoc Tran, Scott A. Sisson, and Christopher C. Drovandi. "Likelihood-free inference in high dimensions with synthetic likelihood". In: *Computational Statistics & Data Analysis* 128 (2018), pp. 271–291.

[128] Satoru Otani, Hérve Daniel, Marie-Paule Roisin, and Francis Crepel. "Dopaminergic modulation of long-term synaptic plasticity in rat prefrontal neurons". In: *Cerebral cortex* 13.11 (2003), pp. 1251–1256.

[129] Lorenzo Pacchiardi and Ritabrata Dutta. *Likelihood-Free Inference with Generative Neural Networks via Scoring Rule Minimization*. 2022. DOI: 10.48550/ARXIV.2205.15784.

[130] 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". In: *Nature methods* 15.10 (2018), pp. 805–815.

[131] George Papamakarios and Iain Murray. "Fast ϵ-free Inference of Simulation Models with Bayesian Conditional Density Estimation". In: *Advances in Neural Information Processing Systems 29*. Curran Associates, Inc., 2016, pp. 1028–1036.

[132] George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. "Normalizing Flows for Probabilistic Modeling and Inference". In: *Journal of Machine Learning Research* 22.57 (2021), pp. 1–64.

[133] George Papamakarios, David Sterratt, and Iain Murray. "Sequential Neural Likelihood: Fast Likelihood-free Inference with Autoregressive Flows". In: *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS)*. Vol. 89. Proceedings of Machine Learning Research. PMLR, 2019, pp. 837–848.

[134] Jaimit Parikh, James Kozloski, and Viatcheslav Gurev. "Integration of AI and mechanistic modeling in generative adversarial networks for stochastic inverse problems". In: *arXiv preprint arXiv:2009.08267* (2020).

[135] Emanuel Parzen. "On estimation of a probability density function and mode". In: *The annals of mathematical statistics* 33.3 (1962), pp. 1065–1076.

[136] Christopher Passaglia, Frederick Dodge, Erik Herzog, Scott Jackson, and Robert Barlow. "Deciphering a neural code for vision". In: *Proceedings of the National Academy of Sciences* 94.23 (1997), pp. 12649–12654.

[137] Karl Pearson. "On lines of closes fit to system of points in space, London, E dinb". In: *Dublin Philos. Mag. J. Sci* 2 (1901), pp. 559–572.

[138] JORGE Perdigao, P Lambrechts, and G Vanherle. "Microscopy investigations: techniques, results, limitations". In: *Am. J. Dent* 13 (2000), p. 3D18D.

[139] Astrid G Petzoldt and Stephan J Sigrist. "Synaptogenesis". In: *Current biology* 24.22 (2014), R1076–R1080.

[140] Kim Cuc Pham, David J Nott, and Sanjay Chaudhuri. "A note on approximating ABC-MCMC using flexible classifiers". In: *STAT* 3.1 (2014), pp. 218–227.

[141] Jonathan W. Pillow, Liam Paninski, Valerie J. Uzzell, Eero P. Simoncelli, and E. J. Chichilnisky. "Prediction and Decoding of Retinal Ganglion Cell Responses with a Probabilistic Spiking Model". In: *Journal of Neuroscience* 25.47 (2005), pp. 11003–11013. DOI: 10.1523/JNEUROSCI.3305-05.2005.

[142] Jonathan K Pritchard, Mark T Seielstad, Anna Perez-Lezaun, and Marcus W Feldman. "Population growth of human Y chromosomes: a study of Y chromosome microsatellites." In: *Molecular Biology and Evolution* 16.12 (1999), pp. 1791–1798.

[143] Stefan T Radev, Ulf K Mertens, Andreas Voss, Lynton Ardizzone, and Ullrich Köthe. "BayesFlow: Learning complex stochastic models with invertible neural networks". In: *IEEE Transactions on Neural Networks and Learning Systems* (2020).

[144] Alec Radford, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks". In: *arXiv preprint arXiv:1511.06434* (2015).

[145] Poornima Ramesh, Mohamad Atayi, and Jakob H Macke. "Adversarial training of neural encoding models on population spike trains". In: *NeuroAI Workshop, Neural Information Processing Systems 2019, Vancouver, Canada*. 2019.

[146] Poornima Ramesh, Basile Confavreux, Pedro J. Goncalves, Tim P. Vogels, and Jakob H. Macke. "The first rule of synaptic plasticity: there is no one rule". In: *In preparation*. 2022.

[147] Poornima Ramesh, Jan-Matthis Lueckmann, Jan Boelts, Álvaro Tejero-Cantero, David S. Greenberg, Pedro J. Goncalves, and Jakob H. Macke. "GATSBI: Generative Adversarial Training for Simulation-Based Inference". In: *International Conference on Learning Representations*. 2022.

[148] Rajesh PN Rao and Dana H Ballard. "Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects". In: *Nature neuroscience* 2.1 (1999), pp. 79–87.

[149] CE. Rasmussen and CKI. Williams. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. MIT Press, 2006.

[150] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. "Generative adversarial text to image synthesis". In: *International conference on machine learning*. PMLR. 2016, pp. 1060–1069.

[151] Fred Rieke, David Warland, Rob de Ruyter Van Steveninck, and William Bialek. *Spikes: exploring the neural code*. MIT press, 1999.

[152] G. S. Rodrigues, D. J. Nott, and S. A. Sisson. *Likelihood-free approximate Gibbs sampling*. 2019. arXiv: 1906.04347 [stat.CO].

[153] Murray Rosenblatt. "Remarks on some nonparametric estimates of a density function". In: *The annals of mathematical statistics* (1956), pp. 832–837.

[154] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning representations by back-propagating errors". In: *nature* 323.6088 (1986), pp. 533–536.

[155] Joao Sacramento, Rui Ponte Costa, Yoshua Bengio, and Walter Senn. "Dendritic error backpropagation in deep cortical microcircuits". In: *arXiv preprint arXiv:1801.00062* (2017).

[156] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. *Improved Techniques for Training GANs*. 2016. DOI: 10.48550/ARXIV.1606.03498.

[157] Wolfram Schultz. “Predictive reward signal of dopamine neurons”. In: *Journal of neurophysiology* 80.1 (1998), pp. 1–27.

[158] Wolfram Schultz, Peter Dayan, and P Read Montague. “A neural substrate of prediction and reward”. In: *Science* 275.5306 (1997), pp. 1593–1599.

[159] João D Semedo, Evren Gokcen, Christian K Machens, Adam Kohn, and M Yu Byron. “Statistical methods for dissecting interactions between brain areas”. In: *Current opinion in neurobiology* 65 (2020), pp. 59–69.

[160] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2014. DOI: 10.48550/ARXIV.1409.1556.

[161] Mewa Singh, Mridula Singh, AK Sharma, and BA Krishna. “Methodological considerations in measurement of dominance in primates”. In: *Current Science* (2003), pp. 709–713.

[162] Scott A Sisson, Yanan Fan, and Mark M Tanaka. “Sequential Monte Carlo without likelihoods”. In: *Proceedings of the National Academy of Sciences* 104.6 (2007), pp. 1760–1765.

[163] Matthew A Smith and Adam Kohn. “Spatial and temporal scales of neuronal correlation in primary visual cortex”. In: *Journal of Neuroscience* 28.48 (2008), pp. 12591–12603.

[164] Akash Srivastava, Lazar Valkov, Chris Russell, Michael U Gutmann, and Charles Sutton. “Veegan: Reducing mode collapse in gans using implicit variational learning”. In: *arXiv preprint arXiv:1705.07761* (2017).

[165] Richard B Stein, Andrew S French, and Andrew V Holden. “The frequency response, coherence, and information capacity of two neuronal models”. In: *Biophysical journal* 12.3 (1972), pp. 295–322.

[166] Armen Stepanyants, Patrick R Hof, and Dmitri B Chklovskii. “Geometry and structural plasticity of synaptic connectivity”. In: *Neuron* 34.2 (2002), pp. 275–288.

[167] David Sussillo and Larry F Abbott. “Generating coherent patterns of activity from chaotic neural networks”. In: *Neuron* 63.4 (2009), pp. 544–557.

[168] Richard S Sutton and Andrew G Barto. “Toward a modern theory of adaptive networks: expectation and prediction.” In: *Psychological review* 88.2 (1981).

[169] Simon Tavaré, David J. Balding, R. C. Griffiths, and Peter Donnelly. “Inferring Coalescence Times From DNA Sequence Data”. In: *Genetics* 145.2 (1997).

[170] J G Taylor, L Jäncke, N J Shah, T Nösselt, N Schmitz, M Himmelback, T Kalenscher, and HW Müller-Gärtner. “A three stage model of awareness: formulation and initial experimental support”. In: *Neuroreport* 9.8 (1998), pp. 1787–1792.

[171] Frédéric Theunissen, J Cooper Roddey, Steven Stufflebeam, Heather Clague, and John P Miller. “Information theoretic analysis of dynamical encoding by four identified primary sensory interneurons in the cricket cercal system”. In: *Journal of Neurophysiology* 75.4 (1996), pp. 1345–1364.

[172] Owen Thomas, Ritabrata Dutta, Jukka Corander, Samuel Kaski, and Michael U. Gutmann. “Likelihood-Free Inference by Ratio Estimation”. In: *Bayesian Analysis* (2021), pp. 1–31.

[173] Simon Thorpe, Arnaud Delorme, and Rufin Van Rullen. “Spike-based strategies for rapid processing”. In: *Neural networks* 14.6-7 (2001), pp. 715–725.

[174] Surya T Tokdar and Robert E Kass. “Importance sampling: a review”. In: *Wiley Interdisciplinary Reviews: Computational Statistics* 2.1 (2010), pp. 54–60.

[175] Susumu Tonegawa, Xu Liu, Steve Ramirez, and Roger Redondo. "Memory engram cells have come of age". In: *Neuron* 87.5 (2015), pp. 918–931.

[176] Tina Toni, David Welch, Natalja Strelkowa, Andreas Ipsen, and Michael PH Stumpf. "Approximate Bayesian computation scheme for parameter inference and model selection in dynamical systems". In: *Journal of the Royal Society Interface* 6.31 (2009), pp. 187–202.

[177] Marcus A. Triplett and Geoffrey J. Goodhill. "Probabilistic Encoding Models for Multivariate Neural Data". In: *Frontiers in Neural Circuits* 13.January (2019).

[178] George Tucker, Andriy Mnih, Chris J. Maddison, Dieterich Lawson, and Jascha Sohl-Dickstein. "REBAR: Low-variance, unbiased gradient estimates for discrete latent variable models". In: *NeurIPS*. 2017, pp. 1–17. arXiv: 1703.07370.

[179] Danil Tyulmankov, Guangyu Robert Yang, and L.F. Abbott. "Meta-learning synaptic plasticity and memory addressing for continual familiarity detection". In: *Neuron* 110.3 (2022), 544–557.e8.

[180] Nachum Ulanovsky, Liora Las, and Israel Nelken. "Processing of low-probability sounds by cortical neurons". In: *Nature neuroscience* 6.4 (2003), pp. 391–398.

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

[182] Tim P. Vogels, Henning Sprekeler, Claudia Clopath, and Wulfram Gerstner. "Inhibitory Plasticity Balances Excitation and Inhibition in Sensory Pathways and Memory Networks". In: *Science* 334 (2011).

[183] Steven S Vogt, Matthew Radovan, Robert Kibrick, R Paul Butler, Barry Alcott, Steve Allen, Pamela Arriagada, Mike Bolte, Jennifer Burt, Jerry Cabak, et al. "APF—The Lick Observatory Automated Planet Finder". In: *Publications of the Astronomical Society of the Pacific* 126.938 (2014), p. 359.

[184] Clarissa L Waites, Ann Marie Craig, and Craig C Garner. "Mechanisms of vertebrate synaptogenesis". In: *Annual review of neuroscience* 28.1 (2005), pp. 251–274.

[185] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. "High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs". In: *CoRR* abs/1711.11585 (2017). arXiv: 1711.11585.

[186] Ronald J. Williams. "Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning". In: *Machine Learning* 8 (1992), pp. 229–256.

[187] Jelmer M. Wolterink, Tim Leiner, Max A. Viergever, and Ivana Išgum. "Generative Adversarial Networks for Noise Reduction in Low-Dose CT". In: *IEEE Transactions on Medical Imaging* 36.12 (2017), pp. 2536–2545. DOI: 10.1109/TMI.2017.2708987.

[188] Simon N Wood. "Statistical inference for noisy nonlinear ecological dynamic systems". In: *Nature* 466.7310 (2010), pp. 1102–1104.

[189] Jiajun Wu, Chengkai Zhang, Tianfan Xue, William T. Freeman, and Joshua B. Tenenbaum. *Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling*. 2016. DOI: 10.48550/ARXIV.1610.07584.

[190] Ruihuang Xie and Fei-Fei Jin. "Two leading ENSO modes and El Niño types in the Zebiak–Cane model". In: *Journal of Climate* 31.5 (2018), pp. 1943–1962.

[191] Liu Yang, Dongkun Zhang, and George Em Karniadakis. *Physics-Informed Generative Adversarial Networks for Stochastic Differential Equations*. 2018. DOI: 10.48550/ARXIV.1811.02033.

[192] Haishan Yao and Yang Dan. "Stimulus timing-dependent plasticity in cortical processing of orientation". In: *Neuron* 32.2 (2001), pp. 315–323.

[193] Byron M Yu, John P Cunningham, Gopal Santhanam, Stephen Ryu, Krishna V Shenoy, and Maneesh Sahani. "Gaussian-process factor analysis for low-dimensional single-trial analysis of neural population activity". In: *Advances in neural information processing systems* 21 (2008).

[194] Manzil Zaheer, Chun-Liang Li, Barnabás Póczos, and Ruslan Salakhutdinov. "GAN Connoisseur: Can GANs Learn Simple 1D Parametric Distributions?" In: *Bridging Theory and Practice Workshop at Neural Information Processing Systems* (2017).

[195] Stephen E Zebiak and Mark A Cane. "A model El-Niño Southern Oscillation". In: *Monthly Weather Review* 115.10 (1987), pp. 2262–2278.

[196] Friedemann Zenke, Everton J Agnes, and Wulfram Gerstner. "Diverse synaptic plasticity mechanisms orchestrated to form and retrieve memories in spiking neural networks". In: *Nature Communications* 6 (2015).

[197] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. "Self-attention generative adversarial networks". In: *International conference on machine learning*. PMLR. 2019, pp. 7354–7363.

[198] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris N Metaxas. "Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 5907–5915.

[199] Le Zhang, Ali Gooya, and Alejandro F Frangi. "Semi-supervised assessment of incomplete LV coverage in cardiac MRI using generative adversarial nets". In: *International workshop on simulation and synthesis in medical imaging*. Springer. 2017, pp. 61–68.

[200] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. *Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks*. 2017. DOI: 10.48550/ARXIV.1703.10593.

Appendices

Adversarial Training of Neural Encoding Models on Population Spike Trains

Poornima Ramesh
poornima.ramesh@tum.de

Mohamad Atayi
bmeatayi@gmail.com

Jakob H. Macke
macke@tum.de

Computational Neuroengineering, Technical University of Munich, Germany

Abstract

Neural population responses to sensory stimuli can exhibit both nonlinear stimulus-dependence and richly structured shared variability. Here, we show how adversarial training can be used to optimize neural encoding models to capture both the deterministic and stochastic components of neural population data. To account for the discrete nature of neural spike trains, we use and compare gradient estimators for adversarial optimization of neural encoding models. We illustrate our approach on population recordings from primary visual cortex. We show that adding latent noise-sources to a convolutional neural network yields a model which captures both the stimulus-dependence and noise correlations of the population activity.

1 Introduction

Neural population activity contains both nonlinear stimulus-dependence and richly structured neural variability. An important challenge for neural encoding models is to generate spike trains that match the statistics of experimentally measured neural population spike trains. Such synthetic spike trains can be used to explore limitations of a model, or as realistic inputs for simulation or stimulation experiments. Most encoding models either focus on modelling the relationship between stimuli and mean-firing rates e.g. [1–3], or on the statistics of correlated variability ('noise correlations'), e.g. [4–6]. They are typically fit with likelihood-based approaches (e.g. maximum likelihood estimation MLE, or variational methods for latent variable models). While this approach is very flexible and powerful, it has mostly been applied to simple models of variability (e.g. Gaussian inputs). Furthermore, MLE-based models are not guaranteed to yield synthetic data that matches the statistics of empirical data, particularly in the presence of latent variables.

Generative adversarial networks (GANs) [7] are an alternative to fitting the parameters of probabilistic models. In adversarial training, the objective is to find parameters which match the statistics of empirical data, using a pair of competing neural networks – a generator and discriminator. The generator maps the distribution of some input random variable onto the empirical data distribution to try and fool the discriminator. The discriminator attempts to classify input data as samples from the true data distribution or from the generator. This approach has been used extensively to produce realistic images [8] and for text generation [9]. Recently, Molano-Mazon et al. [10] trained a generative model of spike trains, and Arakaki et al. [11], rate models of neural populations, using GANs. However, to the best of our knowledge, adversarial training has not yet been used to train spiking models which produce discrete outputs and which aim to capture both the stimulus-dependence of firing rates and shared variability.

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

We propose to use conditional GANs [12] for training neural encoding models, as an alternative to likelihood-based approaches. A key difficulty in using GANs for neural population data is the discrete nature of neural spike trains: Adversarial training requires calculation of gradients through the generative model, which is not possible for models with a discrete sampling step, and hence, requires the application of gradient estimators. While many applications of discrete GANs use biased gradient estimators based on the concrete relaxation technique [13], we find that unbiased gradient estimators REINFORCE[14] and REBAR [15] lead to better fitting performance. We demonstrate our approach by fitting a convolutional neural network model with shared noise sources to multi-electrode recordings from V1 [16].

2 Methods

We want to train a GAN, conditioned on the visual input, to generate multivariate binary spike counts y which match the statistics of empirical data. We model binary spike trains (i.e. each bin t, neuron n and trial i corresponds to an independent draw from a Bernoulli distribution) conditioned on the stimulus x and latent variable z which induces shared variability. We use a convolutional neural network (CNN) f with parameters θ to capture the mapping from x and z to the firing rate λ (Fig. 1), with a sigmoid nonlinearity σ in the last layer,

$$\lambda = \sigma(f(z, x, \theta)) \tag{1}$$

$$y|\lambda \sim \text{Bernoulli}(\lambda). \tag{2}$$

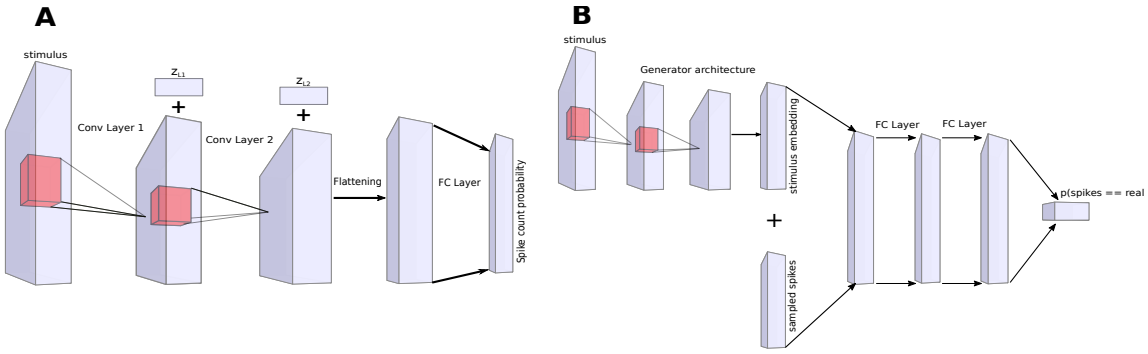

Figure 1: (A) Generator with added noise z_{L1} and z_{L2} (B) Discriminator

The discriminator $\mathcal{D}$ (Fig. 1) receives both the stimulus and the corresponding (simulated or experimental) spike trains. It uses a CNN (similar in architecture to the generator) to embed the stimulus, and combines it with the spike train via fully connected layers. For each timebin and trial, $\mathcal{D}$ outputs the probability of the input spike train being real.

GAN Training The objective is to find the Nash equilibrium of a minmax game between the generator $\mathcal{G}$ and the discriminator $\mathcal{D}$,

$$\min_{\mathcal{G}} \max_{\mathcal{D}} \mathcal{L}(\mathcal{D}, \mathcal{G}) = \mathbb{E}_{y \sim p(y)}[\log\ \mathcal{D}(y|x)] + \mathbb{E}_{z \sim q(z)}[\log(1 - \mathcal{D}(\mathcal{G}(z)|x)). \tag{3}$$

Due to this objective, GANs are notoriously challenging to train as the training algorithm is sensitive to the gradients with respect to the discriminator parameters. We used the cross-entropy objective as in equation 3, but constrained the discriminator gradients using spectral normalisation [17], and employed gradient norm clipping for the generator gradients.

Dealing with discrete data Obtaining the gradients for the generator requires backpropagation through both generator and discriminator networks. Most applications of GANs have been on continuous data. However, spikes are discrete and thus, the generator has a discrete sampling step which blocks backpropagation. Previous attempts to overcome this problem include using concrete relaxation [13], REINFORCE [14] or REBAR [15] to estimate gradients. Concrete relaxation approximates the binary variables as continuous values which are close to 0 and 1. This allows backpropagation through the sampling step, but leads to biased gradients. The REINFORCE gradient estimator provides unbiased but high-variance gradients using the log-derivative trick. The REBAR

gradient estimator combines concrete relaxation and REINFORCE by using the relaxed outputs as a control variate to the REINFORCE gradient. For the applications and hyper-parameter settings we used, we found that both REINFORCE and REBAR performed better than concrete relaxation, and that REBAR did not have much improvement over REINFORCE (Fig. 2).

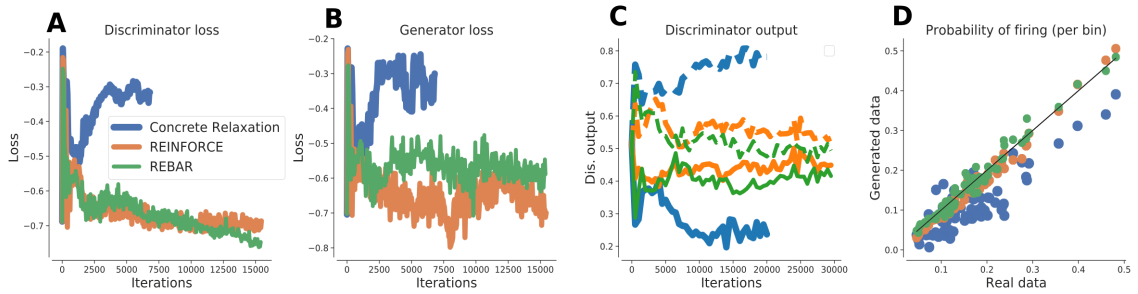

Figure 2: Comparison of different gradient estimation methods for GAN on experimental data. (A) Discriminator loss for concrete relaxation (blue), REINFORCE (orange) and REBAR (green) across training epochs. (B) Generator loss for the 3 gradient estimation methods across training epochs. (C) Discriminator output for real data (dashed lines) and generated data (solid lines) from GANs trained with the 3 gradient estimation methods. (D) Firing rate probability (per bin) calculated on output from GAN generators trained with 3 gradient estimation methods.

Architecture and dataset We fit the GANs to a publicly available dataset [16], consisting of 69 cells recorded from macaque V1, while animals watched a 30s movie repeated 120 times. The movie consisted of 750 frames of size 320 x 320 pixels, which we downsampled to 27 x 27 pixels. We binned the spikes at a 40ms resolution, and binarized the resulting spike trains. Since only 5% of the spike counts in each bin were non-binary after re-binning, we assumed that the binarization would not significantly alter the results.

For the generator, which receives 10 consecutive movie frames of size 27 x 27 pixels as input, we used a 3-layer CNN architecture similar to Kindel et al. [18] (Fig. 1A) – layers 1+2: convolutional with 16 and 32 filters, size 7 by 7, each followed by a MaxPool layer with kernel size 3 and stride 2, followed by LeakyRELUs with slope 0.2. The final layer of the CNN was a fully connected layer with units equal to the number of neurons in the dataset. To capture the stimulus-independent variability shared between the neurons, we added Gaussian white noise to the units of the convolutional layers. The noise was shared between the units of these layers, multiplied by a separate weight for each unit. The discriminator network consisted of a CNN embedding for the stimulus, similar in structure to the generator, but without the added shared noise (Fig. 1B), and 5 fully connected ReLU layers.

Training We trained the two networks in parallel for 15k epochs, each consisting of 2 discriminator and 1 generator update. With batch size 50, we used ADAM with learning rate 0.0001, $\beta_1 = 0.9$ and $\beta_2 = 0.999$ to optimise the network parameters. The first 650 timebins were used for training the networks and the last 100 timebins for validation. All hyper-parameters were set by hand.

3 Results

We fit a 3-layer CNN generative model of binary spike counts to neural population data recorded in the V1 area of macaque visual cortex [16], using adversarial training as described above. For comparison, we fit a CNN with a similar architecture to the GAN generator – but without the shared noise layers – to the same dataset, using supervised learning, i.e. by optimizing the cross-entropy between predicted firing probabilities and experimentally observed spike trains. We also fit a Dichotomised Gaussian (DG) model [5, 19], which explicitly represents shared variability via the covariance matrix of a multi-variate Gaussian distribution.

On the training data, all approaches reproduced the gross structure in the spike train rasters (Fig. 3A) and accurately captured the firing rates (here: spike-probabilities per bin, Fig. 3B left). However, the supervised model did not accurately reproduce total pairwise correlations between the neurons (Fig. 3B center), since its noise-correlations 3C) are constrained to be 0. Thus, the histogram of population spike counts for data generated from the supervised model is also substantially different from that of

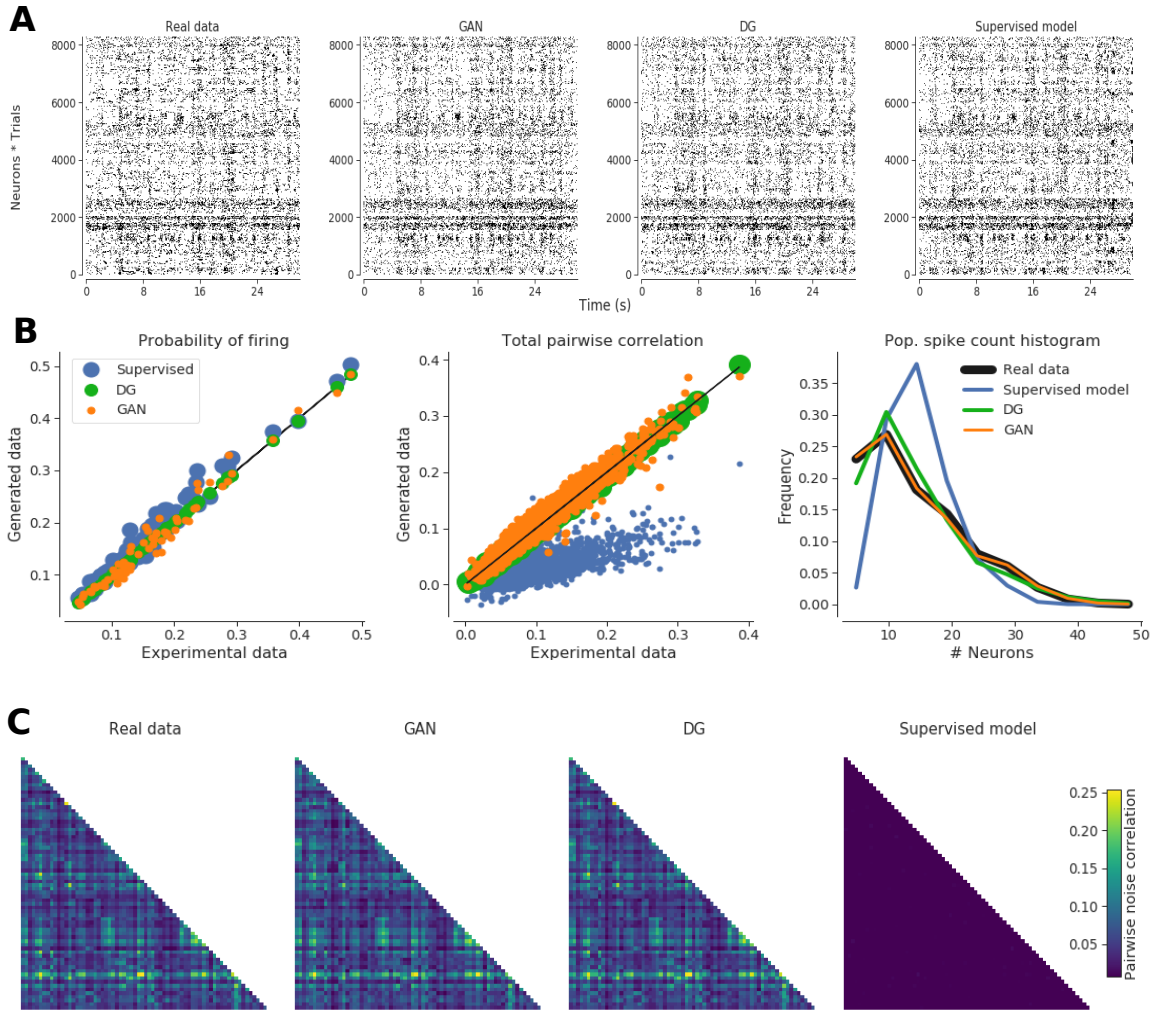

Figure 3: Model comparison on experimental training dataset. (A) Spike train rasters for data and models. (B) Firing rates (per bin, left) and total pairwise correlation (middle) of supervised (blue), DG (green) and GAN generator (orange) model versus data. Right: population spike count histogram for data (black), supervised (blue), DG (green) and GAN generator (orange) model. (C) Pairwise noise correlation matrix for data and models.

the real data (Fig. 3B right). The DG model accurately captured the total correlations and the pairwise noise-correlation matrix since it designed to match the peri-stimulus time histogram (PSTH) and the noise-correlations of the spike trains. However, it did not perfectly capture the population spike count histogram, as it only models second order correlation between neurons, while the population spike count histogram also depends on higher order correlations. In contrast, the GAN generator, with the addition of just a few shared noise parameters to the supervised model, was able to accurately capture the total correlation, the population spike histogram and pairwise noise-correlation matrix.

However, we found that neither model generalised well to the held-out test-dataset, possibly because of the short training-set and high variability of this dataset [16]. When we fit a CNN with the exact same architecture to simulated data with higher SNR and the same dimensions as the V1 dataset, we found that the CNN was able to capture the PSTH and SNR in the test data (Fig. 4).

On the V1 dataset, the adversarially trained CNN and the DG model were similarly good in reproducing correlations and spike count histograms. This might occur if the spike trains of the neural population are homoscedastic (i.e. the variability does not depend on time or stimuli), as assumed by the DG model. Adversarial training is limited only by the flexibility of the network, and can also capture heteroscedastic, i.e. stimulus-dependent noise. Hence, we simulated heteroscedastic data with the same dimensionality as the V1 dataset. We added latent noise to the second layer of the CNN, with variance proportional to the squared input from the previous layer. We fit both a CNN and a DG model to this simulated dataset. We found that CNN trained with the GAN framework was

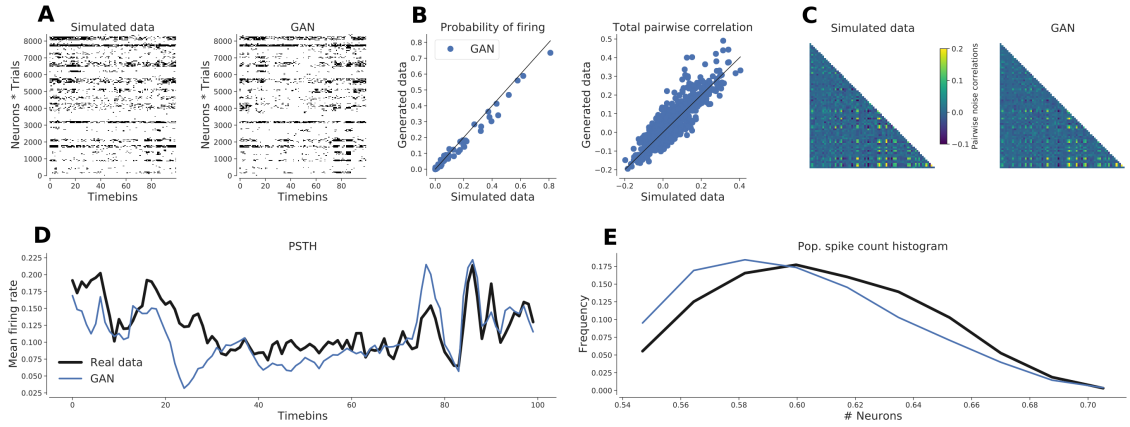

Figure 4: GAN generator fit on simulated test dataset. (A) Spike train rasters for test data and model. (B) Firing rates (per bin, left) and total pairwise correlation (right) of GAN generator model versus data. (C) Pairwise noise correlation matrix for data and model. (D) PSTH averaged across trials and neural population for data(black) and model(blue). (E) Population spike count histogram for data (black) and model (blue).

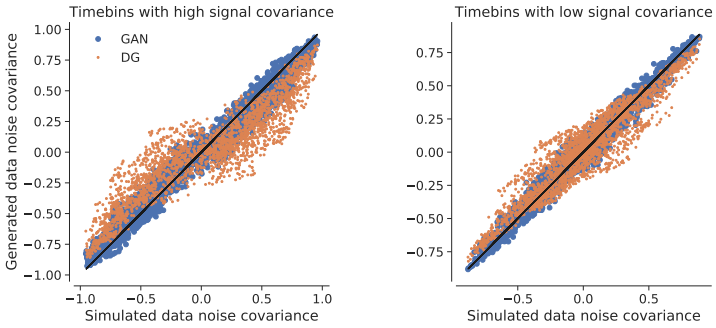

Figure 5: Pairwise noise covariance for simulated heteroscedastic dataset Noise covariance for simulated data versus GAN (blue) and DG (orange) models in high signal covariance timebins(left) and low signal covariance timebins (right).

able to accurately capture the covariances estimated from 'low signal' and 'high signal' timebins separately (Fig. 5), unlike the DG model.

4 Discussion

We here showed how adversarial training of conditional generative models that produce discrete outputs (i.e. neural spike trains) can be used to generate data that matches the distribution of spike trains recorded in-vivo, and in particular, its firing rates and correlations. We used unbiased gradient estimators to train conditional GANs on discrete spike trains and spectral normalisation to stabilise training. However, training of discrete GANs remains sensitive to the architecture of the discriminator, as well as hyper-parameter settings. We showed that we are able to successfully train adversarial models in cases where supervised and Dichotomised Gaussian models fail.

In future, adversarial training could be used to capture higher-order structure in neural data, and could be combined with discriminators that target certain statistics of the data that might be of particular interest, in a spirit similar to maximum entropy models [4]. Similarly, this approach could also be extended to capture temporal features in neural population data [20] such as spike-history dependence or adaptation effects. Since we condition the discriminator on the input stimulus, adversarial training could be used for transfer learning across multiple datasets. Generative models trained this way to produce realistic spike trains to various input stimuli, may be used to probe the range of spiking behaviour in a neural population under different kinds of stimulus or noise perturbations.

Acknowledgements

We thank all members of CNE, especially David Greenberg and Artur Speiser, for feedback and discussions. We thank Matthew Smith and Adam Kohn for sharing data via CRCNS.org. Funding was provided by the DFG through SFB 1233 (276693517).

References

J A Nelder and R W M Wedderburn. Generalized Linear Models. *J. R. Statist. Soc. A.*, 135(3):370–384, 1972. ISSN 0162-1459.

Lane McIntosh, Niru Maheswaranathan, Aran Nayebi, Surya Ganguli, and Stephen Baccus. Deep learning models of the retinal response to natural scenes. *Advances in neural information processing systems*, 29, 02 2017.

Santiago Cadena, George Denfield, Edgar Walker, Leon Gatys, Andreas Tolias, Matthias Bethge, and Alexander Ecker. Deep convolutional models improve predictions of macaque v1 responses to natural images. *PLOS Computational Biology*, 15:e1006897, 04 2019. doi: 10.1371/journal.pcbi.1006897.

Elad Schneidman, Michael J Berry, Ronen Segev, and William Bialek. Weak pairwise correlations imply strongly correlated network states in a neural population. *Nature*, 440(7087):1007–12, 2006. ISSN 1476-4687.

Jakob H Macke, Philipp Berens, Alexander S Ecker, Andreas S Tolias, and Matthias Bethge. Generating Spike Trains with Specified Correlation Coefficients. *Neural Computation*, 423:397–423, 2009.

Evan W Archer, Urs Koster, Jonathan W Pillow, and Jakob H Macke. Low-dimensional models of neural population activity in sensory cortical circuits. In *NeurIPS*, pages 343–351, 2014.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*, 2018.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. *ArXiv*, 2016. ISSN 1525-822X.

Manuel Molano-Mazon, Arno Onken, Eugenio Piasini, and Stefano Panzeri. Synthesizing realistic neural population activity patterns using Generative Adversarial Networks. *ICLR*, 2018.

Takafumi Arakaki, G. Barello, and Yashar Ahmadian. Inferring neural circuit structure from datasets of heterogeneous tuning curves. *PLoS Computational Biology*, 15(4), 2019.

Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv:1411.1784*, 2014.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: a continuous relaxation of discrete random variables. *ArXiv*, 2016. ISSN 1528-4336.

Ronald J. Williams. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. *Machine Learning*, (8):229–256, 1992. ISSN 18736513.

George Tucker, Andriy Mnih, Chris J. Maddison, Dieterich Lawson, and Jascha Sohl-Dickstein. REBAR: Low-variance, unbiased gradient estimates for discrete latent variable models. In *NeurIPS*, pages 1–17, 2017.

A. Kohn and M. A. Smith. *Utah array extracellular recordings of spontaneous and visually evoked activity from anesthetized macaque primary visual cortex (V1)*. CRCNS.org, 2016.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *CoRR*, abs/1802.05957, 2018.

William F. Kindel, Elijah D. Christensen, and Joel Zylberberg. Using deep learning to probe the neural code for images in primary visual cortex. *Journal of Vision*, 19(4):29–29, 04 2019. ISSN 1534-7362. doi: 10.1167/19.4.29. URL `https://doi.org/10.1167/19.4.29`.

Dmitry R. Lyamzin, Jakob H. Macke, and Nicholas A. Lesica. Modeling Population Spike Trains with Specified Time-Varying Spike Rates, Trial-to-Trial Variability, and Pairwise Signal and Noise Correlations. *Frontiers in Computational Neuroscience*, 4(November):1–11, 2010. ISSN 1662-5188.

Jakob H Macke, L Büsing, John P Cunningham, B M Yu, KV Shenoy, and M Sahani. Empirical models of spiking in neural populations. In *NeurIPS*, pages 1350–1358, 2011.

GATSBI: GENERATIVE ADVERSARIAL TRAINING FOR SIMULATION-BASED INFERENCE

Poornima Ramesh
University of Tübingen

Jan-Matthis Lueckmann
University of Tübingen

Jan Boelts
TU Munich

Álvaro Tejero-Cantero
University of Tübingen

David S. Greenberg
Helmholtz Centre Hereon

Pedro J. Gonçalves
University of Tübingen

Jakob H. Macke
University of Tübingen

ABSTRACT

Simulation-based inference (SBI) refers to statistical inference on stochastic models for which we can generate samples, but not compute likelihoods. Like SBI algorithms, generative adversarial networks (GANs) do not require explicit likelihoods. We study the relationship between SBI and GANs, and introduce GATSBI, an adversarial approach to SBI. GATSBI reformulates the variational objective in an adversarial setting to learn implicit posterior distributions. Inference with GATSBI is amortised across observations, works in high-dimensional posterior spaces and supports implicit priors. We evaluate GATSBI on two SBI benchmark problems and on two high-dimensional simulators. On a model for wave propagation on the surface of a shallow water body, we show that GATSBI can return well-calibrated posterior estimates even in high dimensions. On a model of camera optics, it infers a high-dimensional posterior given an implicit prior, and performs better than a state-of-the-art SBI approach. We also show how GATSBI can be extended to perform sequential posterior estimation to focus on individual observations. Overall, GATSBI opens up opportunities for leveraging advances in GANs to perform Bayesian inference on high-dimensional simulation-based models.

1 INTRODUCTION

Hypothesis-making in many scientific disciplines relies on stochastic simulators that—unlike expressive statistical models such as neural networks—have domain-relevant, interpretable parameters. Finding the parameters θ of a simulator that reproduce the observed data x_o constitutes an inverse problem. In order to use these simulators to formulate further hypotheses and experiments, one needs to obtain uncertainty estimates for the parameters and allow for multi-valuedness, i.e., different candidate parameters accounting for the same observation. These requirements are met by Bayesian inference, which attempts to approximate the posterior distribution $p(\theta|x_\mathrm{o})$. While a variety of techniques exist to calculate posteriors for scientific simulators which allow explicit likelihood calculations $p(x|\theta)$, inference on black-box simulators with intractable likelihoods a.k.a. 'simulation-based inference' (Cranmer et al., 2020), poses substantial challenges. In traditional, so-called Approximate Bayesian Computation (ABC) approaches to SBI (Beaumont et al., 2002, 2009; Marjoram et al., 2003; Sisson et al., 2007), one samples parameters θ from a prior $\pi(\theta)$ and accepts only those parameters for which the simulation output $x \sim p(x|\theta)$ is close to the observation $d(x, x_\mathrm{o}) < \epsilon$. With increasing data dimensionality N, this approach incurs an exponentially growing simulation expense ('curse of dimensionality'); it also requires suitable choices of distance function d and acceptance threshold ϵ.

Recent SBI algorithms (e.g., Greenberg et al., 2019; Gutmann and Corander, 2015; Hermans et al., 2020; Lueckmann et al., 2017; Meeds and Welling, 2014; Papamakarios and Murray, 2016; Radev et al., 2020; Thomas et al., 2021) draw on advances in machine learning and are often based on Gaussian Processes (Rasmussen and Williams, 2006) or neural networks for density estimation (e.g. using normalizing flows; Papamakarios et al., 2021). While this has led to substantial improvements over classical approaches, and numerous applications in fields such as cosmology (Cole et al., 2021), neuroscience (Gonçalves et al., 2020) or robotics (Muratore et al., 2021), statistical inference for

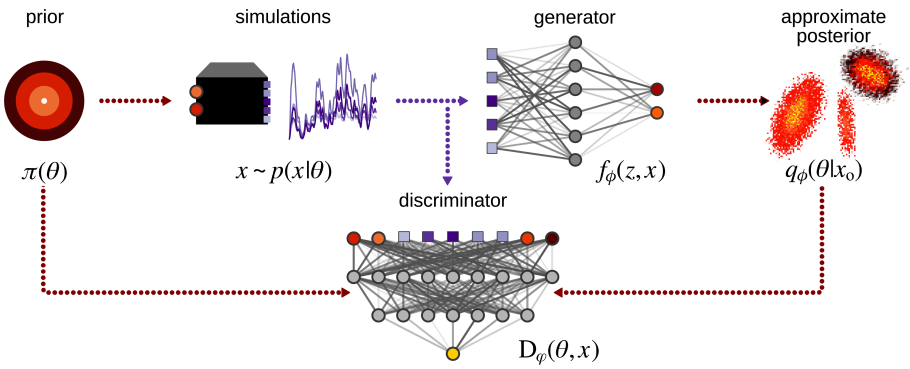

Figure 1: **GATSBI uses a conditional generative adversarial network (GAN) for simulation-based inference (SBI)**. We sample parameters θ from a prior $\pi(\theta)$, and use them to generate synthetic data x from a black-box simulator. The GAN generator learns an implicit approximate posterior $q_\phi(\theta|x)$, i.e., it learns to generate posterior samples θ' given data x. The discriminator is trained to differentiate between θ' and θ, conditioned on x.

high-dimensional parameter spaces is an open challenge (Cranmer et al., 2020; Lueckmann et al., 2021): This prevents the application of these methods to many real-world scientific simulators.

In machine learning, generative adversarial networks (GANs, Goodfellow et al., 2014) have emerged as powerful tools for learning generative models of high-dimensional data, for instance, images (Isola et al., 2016; Radford et al., 2015; Zhu et al., 2017). Using a minimax game between a 'generator' and a 'discriminator', GANs can generate samples which look uncannily similar to the data they have been trained on. Like SBI algorithms, adversarial training is 'likelihood-free', i.e., it does not require explicit evaluation of a density function, and also relies on comparing empirical and simulated data. This raises the question of whether GANs and SBI solve the same problem, and in particular whether and how adversarial training can help scale SBI to high-dimensional regimes.

Here, we address these questions and make the following contributions: First, we show how one can use adversarial training to learn an implicit posterior density for SBI. We refer to this approach as GATSBI (Generative Adversarial Training for SBI)[1]. Second, we show that GATSBI can be formulated as an adversarial variational inference algorithm (Huszár, 2017; Makhzani et al., 2015; Mescheder et al., 2017), thus opening up a potential area of exploration for SBI approaches. Third, we show how it can be extended to refine posterior estimates for specific observations, i.e., sequential posterior estimation. Fourth, on two low-dimensional SBI benchmark problems, we show that GATSBI's performance is comparable to common SBI algorithms. Fifth, we illustrate GATSBI on high-dimensional inference problems which are out of reach for most SBI methods: In a fluid dynamics problem from earth science, we learn an implicit posterior over 100 parameters. Furthermore, we show that GATSBI (in contrast to most SBI algorithms) can be used on problems in which the *prior* is only specified implicitly: by recasting image-denoising as an SBI problem, we learn a posterior over (784-dimensional) images. Finally, we discuss the opportunities and limitations of GATSBI relative to current SBI algorithms, and chart out avenues for future work.

2 Methods

2.1 Simulation-based inference and adversarial networks

Simulation-based inference (SBI) targets stochastic simulators with parameters θ that we can use to generate samples x. In many scientific applications, simulators are 'black-box': we cannot evaluate the likelihood $p(x|\theta)$, take derivatives through the simulator, or access its internal random numbers. The goal of SBI is to infer the parameters θ, given empirically observed data x_o i.e., to obtain the posterior distribution $p(\theta|x_o)$. In most applications of existing SBI algorithms, the simulator is a mechanistic forward model depending on a low (~ 10) number of parameters θ.

[1]Note that the name is shared by recently proposed, and entirely unrelated, algorithms (Min et al., 2021; Yu et al., 2020)

Generative adversarial networks (GANs) (Goodfellow et al., 2014) consist of two networks trained adversarially. The generator $f_\phi(z)$ produces samples x from latent variables z; the discriminator D_ψ acts as a similarity measure between the distribution of generated x and observations x_o from a distribution $p_\mathrm{data}(x_\mathrm{o})$. The two networks are pitted against each other: the generator attempts to produce samples that would fool the discriminator; the discriminator is trained to tell apart generated and ground-truth samples. This takes the form of a minimax game $\min_\phi \max_\psi L(\phi, \psi)$ between the networks, with cross-entropy as the value function:

$$L(\phi, \psi) = \mathbb{E}_{p_\mathrm{data}(x_\mathrm{o})} \log \mathrm{D}_\psi(x_\mathrm{o}) + \mathbb{E}_{p(z)} \log(1 - \mathrm{D}_\psi(f_\phi(z))).$$

At the end of training, if the networks are sufficiently expressive, samples from f_ϕ are hard to distinguish from the ground-truth samples, and the generator approximates the observed data distribution $p_\mathrm{data}(x_\mathrm{o})$. GANs can also be used to perform (implicit) *conditional* density estimation, by making the generator and discriminator conditional on external covariates t (Mirza and Osindero, 2014). Note that the GAN formulation does not evaluate the intractable distribution $p_\mathrm{data}(x_\mathrm{o})$ or $p_\mathrm{data}(x_\mathrm{o}|t)$ at any point. Thus, GANs and SBI appear to be solving similar problems: They are both 'likelihood-free' i.e., they do not evaluate the target distribution, but rather exclusively work with samples from it.

Nevertheless, GANs and SBI differ in multiple ways: GANs use neural networks, and thus are always differentiable, whereas the numerical simulators in SBI may not be. Furthermore, the latent variables for the GAN generator are typically accessible, while in SBI, one may not have access to the simulator's internal random variables. Finally, in GANs, the goal is to match the distribution of the generated samples x to that of the observed data x_o, and parameters of the generator are learnt as point estimates. In contrast, the goal of SBI is to learn a distribution over parameters θ of a simulator consistent with both the observed data x_o *and* prior knowledge about the parameters. However, conditional GANs *can* be used to perform conditional density estimation. How can we repurpose them for SBI? We propose an approach using adversarial training for SBI, by employing GANs as implicit density estimators rather than generative models of data.

2.2 Generative Adversarial Training for SBI: GATSBI

Our approach leverages conditional GANs for SBI. We train a deep generative neural network f_ϕ with parameters ϕ, i.e., f takes as inputs observations x and noise z from a fixed source with distribution $p(z)$, so that $f_\phi(x, z)$ deterministically transforms z to generate θ. This generative network f is an implicit approximation $q_\phi(\theta|x)$ of the posterior distribution, i.e.,

$$\left.\begin{array}{l} z \sim p(z) \\ \theta = f_\phi(z, x) \end{array}\right\} \Rightarrow \theta \sim q_\phi(\theta|x). \tag{1}$$

We train f adversarially: We sample many parameters θ from the prior, and for each θ, simulate $x \sim p(x|\theta)$ i.e., we sample tuples $(\theta, x) \sim \pi(\theta)p(x|\theta) = p(\theta|x)p(x)$. These samples from the joint constitute our training data. We define a discriminator D, parameterised by ψ, to differentiate between samples θ drawn from the approximate posterior $q_\phi(\theta|x)$ and the true posterior $p(\theta|x)$. The discriminator is conditioned on x. We calculate the cross-entropy of the discriminator outputs, and maximize it with respect to the discriminator parameters ψ and minimize it with respect to the generator parameters ϕ i.e., $\min_\phi \max_\psi L(\phi, \psi)$, with

$$\begin{aligned} L(\phi, \psi) &= \mathbb{E}_{\pi(\theta)p(x|\theta)p(z)} \Big[\log \mathrm{D}_\psi(\theta, x) + \log \big(1 - \mathrm{D}_\psi(f_\phi(z, x), x)\big) \Big] \\ &= \mathbb{E}_{p(x)} \Big[\mathbb{E}_{p(\theta|x)} \Big(\log \mathrm{D}_\psi(\theta, x) \Big) + \mathbb{E}_{q(\theta|x)} \Big(\log \big(1 - \mathrm{D}_\psi(\theta, x)\big) \Big) \Big] \end{aligned} \tag{2}$$

This conditional GAN value function targets the desired posterior $p(\theta|x)$:

Proposition 1. *Given a fixed generator f_ϕ, the discriminator D_{ψ^*} maximizing equation 2 satisfies*

$$\mathrm{D}_{\psi^*}(\theta, x) = \frac{p(\theta|x)}{p(\theta|x) + q_\phi(\theta|x)}, \tag{3}$$

and the corresponding loss function for the generator parameters is the Jensen-Shannon divergence (JSD) between the true and approximate posterior,

$$L_{\psi^*}(\phi) = 2\,\mathrm{JSD}(p(\theta|x)||q_\phi(\theta|x)) - \log 4.$$

Using sufficiently flexible networks, the GATSBI generator trained with the cross-entropy loss converges in the limit of infinitely many samples to the true posterior, since the JSD is minimised if and only if $q_\phi(\theta|x) = p(\theta|x)$. The proof directly follows Prop. 1 and 2 from Goodfellow et al. (2014) and is given in detail in App. A.1. GATSBI thus performs *amortised inference*, i.e., learns a single inference network which can be used to perform inference rapidly and efficiently for a wide range of potential observations, without having to re-train the network for each new observation. The training steps for GATSBI are described in App. B.

2.3 Relating GATSBI to existing SBI approaches

There have been previous attempts at using GANs to fit black-box scientific simulators to empirical data. However, unlike GATSBI, these approaches do not perform Bayesian inference: Kim et al. (2020) and Louppe et al. (2017) train a discriminator to differentiate between samples from the black-box simulator and empirical observations. They use the discriminator to adjust a proposal distribution for simulator parameters until the resulting simulated data and empirical data match. However, neither approach returns a posterior distribution representing the balance between a prior distribution and a fit to empirical data. Similarly, Jethava and Dubhashi (2017) use two generator networks, one to approximate the prior and one to learn summary statistics, as well as two discriminators. This scheme yields a generator for parameters leading to realistic simulations, but does not return a posterior over parameters. Parikh et al. (2020) train a conditional GAN (c-GAN), as well as two additional GANs (r-GAN and t-GAN) within an active learning scheme, in order to solve 'population of models' problems (Britton et al., 2013; Lawson et al., 2018). While c-GAN exhibits similarities with GATSBI, the method does not do Bayesian inference, but rather solves a constrained optimization problem. Adler and Öktem (2018) use Wasserstein-GANs (Arjovsky et al., 2017) to solve Bayesian inverse problems in medical imaging. While their set-up is similar to GATSBI, the Wasserstein metric imposes stronger conditions for the generator to converge to the true posterior (App. Sec. A.3).

Several SBI approaches train discriminators to circumvent evaluation of the intractable likelihood (e.g., Cranmer et al., 2015; Gutmann et al., 2018; Pham et al., 2014; Thomas et al., 2021). Hermans et al. (2020) train density-ratio estimators by discriminating samples $(\theta, x) \sim \pi(\theta)p(x|\theta)$ from $(\theta, x) \sim \pi(\theta)p(x)$, which can be used to sample from the posterior. Unlike GATSBI, this requires a potentially expensive step of MCMC sampling. A closely related approach (Durkan et al., 2020; Greenberg et al., 2019) directly targets the posterior, but requires a density estimator that can be evaluated—unlike GATSBI, where the only restriction on the generator is that it remains differentiable.

Finally, ABC and synthetic likelihood methods have been developed to tackle problems in high-dimensional parameter spaces (>100-dimensions) although these require model assumptions regarding the Gaussianity of the likelihood (Ong et al., 2018), or low-dimensional summary statistics (Kousathanas et al., 2015; Rodrigues et al., 2019), in addition to MCMC sampling.

2.4 Relation to adversarial inference

GATSBI reveals a direct connection between SBI and adversarial variational inference. Density estimation approaches for SBI usually use the forward Kullback-Leibler divergence (D_{KL}). The reverse D_{KL} can offer complementary advantages (e.g. by promoting mode-seeking rather than mode-covering (Turner and Sahani, 2011)), but is hard to compute and optimise for conventional density estimation approaches. We here show how GATSBI, by using an approach similar to adversarial variational inference, performs an optimization that also finds a minimum of the reverse D_{KL}.

First, we note that density estimation approaches for SBI involve maximising the approximate log posterior over samples simulated from the prior. This is equivalent to minimising the *forward* D_{KL}:

$$L_{\mathrm{fwd}}(\phi) = \mathbb{E}_{p(x)} D_{\mathrm{KL}}(p(\theta|x)||q_\phi(\theta|x)) = -\mathbb{E}_{p(x)p(\theta|x)} \log q_\phi(\theta|x) + \mathbb{E}_{p(x)p(\theta|x)} \log p(\theta|x). \quad (4)$$

rather than reverse D_{KL}:

$$L_{\mathrm{rev}}(\phi) = \mathbb{E}_{p(x)} D_{\mathrm{KL}}(q_\phi(\theta|x)||p(\theta|x)) = \mathbb{E}_{p(x)q_\phi(\theta|x)} \log \frac{q_\phi(\theta|x)}{p(\theta|x)}, \quad (5)$$

This is because it is feasible to compute L_{fwd} with an intractable likelihood: it only involves evaluating the approximate posterior under *samples* from the true joint distribution $p(\theta, x) = p(x|\theta)\pi(\theta)$. The problem with the reverse D_{KL}, however, is that it requires evaluating the true posterior $p(\theta|x)$ under

samples from the approximate posterior $q_\phi(\theta|x)$, which is impossible with an intractable likelihood. This is a problem also shared by variational-autoencoders (VAEs, Kingma and Welling, 2013) but which can be solved using adversarial inference.

A VAE typically consists of an encoder network $q_\phi(u|x)$ approximating the posterior over latents u given data x, and a decoder network approximating the likelihood $p_\alpha(x|u)$. The parameters ϕ and α of the two networks are optimised by maximising the Evidence Lower Bound (ELBO):

$$\begin{aligned}\mathrm{ELBO}(\alpha,\phi) &= -\mathbb{E}_{p(x)}\left[D_{\mathrm{KL}}(q_\phi(u|x)||p(u)) + \mathbb{E}_{q_\phi(u|x)}[\log p_\alpha(x|u)]\right] \\ &= -\mathbb{E}_{p(x)q_\phi(u|x)} \log \frac{q_\phi(u|x)p(x)}{p(u)p_\alpha(x|u)} + \text{const.} \\ &= -D_{\mathrm{KL}}(q_\phi(u|x)p(x)||p(u)p_\alpha(x|u)) + \text{const.} \end{aligned} \tag{6}$$

Note that the D_{KL} is minimised when maximising the ELBO. If any of the distributions used to compute the ratio in the D_{KL} in equation 6 is intractable, one can do adversarial inference, i.e., the ratio can be approximated using a discriminator (see Prop. 1) and $q_\phi(u|x)$ is recast as the generator. Various adversarial inference approaches approximate different ratios, depending on which of the distributions is intractable (see App. Table 1 for a non-exhaustive comparison of different discriminators and corresponding losses), and their connection to GANs has been explored extensively (Chen et al., 2018; Hu et al., 2017; Mohamed and Lakshminarayanan, 2016; Srivastava et al., 2017).

Recasting SBI into the VAE framework, the intractable distribution is the likelihood $p_\alpha(x|\theta)$,[2] with α constant, i.e., the simulator only has parameters to do inference over. Hence, $p(x)$ is defined as the marginal likelihood. In this setting, the SBI objective becomes the reverse D_{KL}, and we can use an adversarial method, e.g. GATSBI, to minimise it for learning $q_\phi(\theta|x)$. There is an additional subtlety at play here: GATSBI's generator loss, given an optimal discriminator, is not the reverse D_{KL}, but the JSD (see Prop. 1). However, following the proof for Prop. 3 in Mescheder et al. (2017), we show that the approximate posterior that minimises the JSD *also* maximises the ELBO (see App. A.2). Thus, GATSBI is an adversarial inference method that performs SBI using the *reverse* D_{KL}.

Note that the converse is not true: not all adversarial inference methods can be adapted for SBI. Specifically, some adversarial inference methods require explicit evaluation of the likelihood. Of those methods listed in Table 1, only likelihood-free variational inference (LFVI) (Tran et al., 2017) is explicitly used to perform inference for a simulator with intractable likelihood. LFVI is defined as performing inference over latents u and global parameters β shared across multiple observations x, which are sampled from an empirical distribution $p'(x) \neq p(x)$, i.e., it learns $q_\phi(u, \beta|x)$. However, if $p'(x) = p(x)$, and β is constant, i.e., the simulator does not have tunable parameters (but only parameters one does inference over) then LFVI and GATSBI become (almost) equivalent: LFVI differs by a single term in the loss function, and we empirically found the methods to yield similar results (see App. Fig. 6). We discuss LFVI's relationship to GATSBI in more detail in App. A.3.

2.5 SEQUENTIAL GATSBI

The GATSBI formulation in the previous sections allows us to learn an amortised posterior $p(\theta|x)$ for *any* observation x. However, one is often interested in good posterior samples for a particular experimental observation x_{o}—and not for all possible observations x. This can be achieved by training the density estimator using samples θ from a proposal prior $\tilde{\pi}(\theta)$ instead of the usual prior $\pi(\theta)$. The proposal prior ideally produces samples θ that are localised around the modes of $p(\theta|x_{\mathrm{o}})$ and can guide the density estimator towards inferring a posterior for x_{o} using less training data. To this end, sequential schemes using proposal distributions to generate training data over several rounds of inference (e.g. by using the approximate posterior from previous rounds as the proposal prior for subsequent rounds) can be more sample efficient (Lueckmann et al., 2021).

Sequential GATSBI uses a scheme similar to other sequential SBI methods (e.g., Greenberg et al., 2019; Hermans et al., 2020; Lueckmann et al., 2017; Papamakarios and Murray, 2016; Papamakarios et al., 2019): the generator from the current round of posterior approximation becomes the proposal distribution in the next round. However, using samples from a proposal prior $\tilde{\pi}(\theta)$ to train the GAN would lead to the GATSBI generator learning a *proposal posterior* $\tilde{p}(\theta|x)$ rather than the true posterior

[2]Note the difference in notation: the parameters θ correspond to the latents u of a VAE, and the simulator in SBI to the decoder of the VAE.

$p(\theta|x)$. Previous sequential methods solve this problem either by post-hoc correcting the learned density (Papamakarios and Murray, 2016), by adjusting the loss function (Lueckmann et al., 2017), or by reparametrizing the posterior network (Greenberg et al., 2019). While these approaches might work for GATSBI (see App. A.4 for an outline), we propose a different approach: at training time, we sample the latents z—that the generator transforms into parameters θ—from a *corrected* distribution $p_t(z)$, so that the samples θ are implicitly generated from an approximate proposal posterior $\tilde{q}_\phi(\theta|x)$:

$$\left.\begin{array}{l} z \sim p_t(z) \\ \theta = f_\phi(z, x) \end{array}\right\} \Rightarrow \theta \sim \tilde{q}_\phi(\theta|x), \tag{7}$$

where $p_t(z) = p(z)\ (\omega(f_\phi(z,x),x))^{-1}$, $\omega(\theta, x) = \frac{\pi(\theta)}{\tilde{\pi}(\theta)} \frac{\tilde{p}(x)}{p(x)}$, and $\tilde{p}(x)$ is the marginal likelihood under samples from the proposal prior. Since $\tilde{\pi}(\theta)$, $p(x)$ and $\tilde{p}(x)$ are intractable, we approximate $\omega(\theta, x)$ using ratio density estimators for $\frac{\pi(\theta)}{\tilde{\pi}(\theta)}$ and $\frac{\tilde{p}(x)}{p(x)}$. This ensures that at inference time, the GATSBI generator transforms samples from $p(z)$ into an approximation of the true posterior $p(\theta|x)$ with highest accuracy at $x = x_\text{o}$ (see App. A.4, B). Azadi et al. (2019); Che et al. (2020) use similar approaches to improve GAN training. Note that this scheme may be used with other inference algorithms evaluating a loss under approximate posterior samples, e.g. LFVI (see Sec. 2.4).

3 Results

We first investigate GATSBI on two common benchmark problems in SBI (Lueckmann et al., 2021). We then show how GATSBI infers posteriors in a high-dimensional regime where most state-of-the-art SBI algorithms are inadequate, as well as in a problem with an implicit prior (details in App. D.1).

3.1 Benchmark problems

We selected two benchmark problems with low-dimensional posteriors on which state-of-the-art SBI algorithms have been tested extensively. We used the benchmarking-tools and setup from Lueckmann et al. (2021). Briefly, we compared GATSBI against five SBI algorithms: classical rejection ABC (Tavaré et al., 1997, REJ-ABC), sequential Monte Carlo ABC (Beaumont et al., 2009, SMC-ABC) and the single-round variants of flow-based neural likelihood estimation (Papamakarios et al., 2019, NLE), flow-based neural posterior estimation (Greenberg et al., 2019; Papamakarios and Murray, 2016, NPE) and neural ratio estimation (Durkan et al., 2020; Hermans et al., 2020, NRE). GANs have previously been shown to be poor density estimators for low-dimensional distributions (Zaheer et al., 2017). Thus, while we include these examples to provide empirical support that GATSBI works, we do not expect it to perform as well as state-of-the-art SBI algorithms on these tasks. We compared samples from the approximate posterior against a reference posterior using a classifier trained on the two sets of samples (classification-based two-sample test, C2ST). A C2ST accuracy of 0.5 (chance level) corresponds to perfect posterior estimation.

"Simple Likelihood Complex Posterior" (SLCP) is a challenging SBI problem designed to have a simple likelihood and a complex posterior (Papamakarios et al., 2019). The prior $\pi(\theta)$ is a 5-dimensional uniform distribution and the likelihood for the 8-dimensional x is a Gaussian whose mean and variance are nonlinear functions of θ. This induces a posterior over θ with four symmetrical modes and vertical cut-offs (see App. Fig. 7), making it a challenging inference problem. We trained all algorithms on a budget of 1k, 10k and 100k simulations. The GATSBI C2ST scores were on par with NRE, better than REJ-ABC and SMC-ABC, but below the NPE and NLE scores (Figure 2A).

The **"Two-Moons Model"** (Greenberg et al., 2019) has a simple likelihood and a bimodal posterior, where each mode is crescent-shaped (see App. Fig. 8). Both x and θ are two-dimensional. We initially trained GATSBI with the same architecture and settings as for the SLCP problem. While the classification score for GATSBI (see Figure 2B) was on par with that of REJ-ABC and SMC-ABC for 1k simulations, performance did not significantly improve when increasing training budget. When we tuned the GAN architecture and training hyperparameters specifically for this model, we found a performance improvement. Lueckmann et al. (2021) showed that performance can be significantly improved with sequential inference schemes. We found that although using sequential schemes with GATSBI leads to small performance improvements, it still did not ensure performance on-par with the flow-based methods (see App. Fig. 9). Moreover, since it had double the number of hyperparameters as amortised GATSBI, sequential GATSBI also required more hyperparameter tuning.

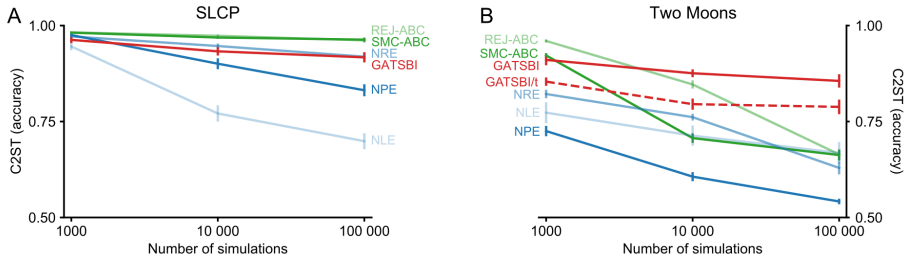

Figure 2: **GATSBI performance on benchmark problems.** Mean C2ST score ($\pm$ standard error of the mean) for 10 observations each. A. The classification score for GATSBI (red) on SCLP decreases with increasing simulation budget, and is comparable to NRE. It outperforms rejection ABC and SMC-ABC, but has worse performance than NPE and NLE. B. The classification score for GATSBI (red) on the two-moons model decreases with increasing simulation budget, is comparable to REJ-ABC and SMC-ABC for simulation budgets of 1k and 10k, and is outperformed by all other methods for a 100k simulation budget. However, GATSBI's classification score improves when its architecture and optimization parameters are tuned to the two-moons model (red, dashed).

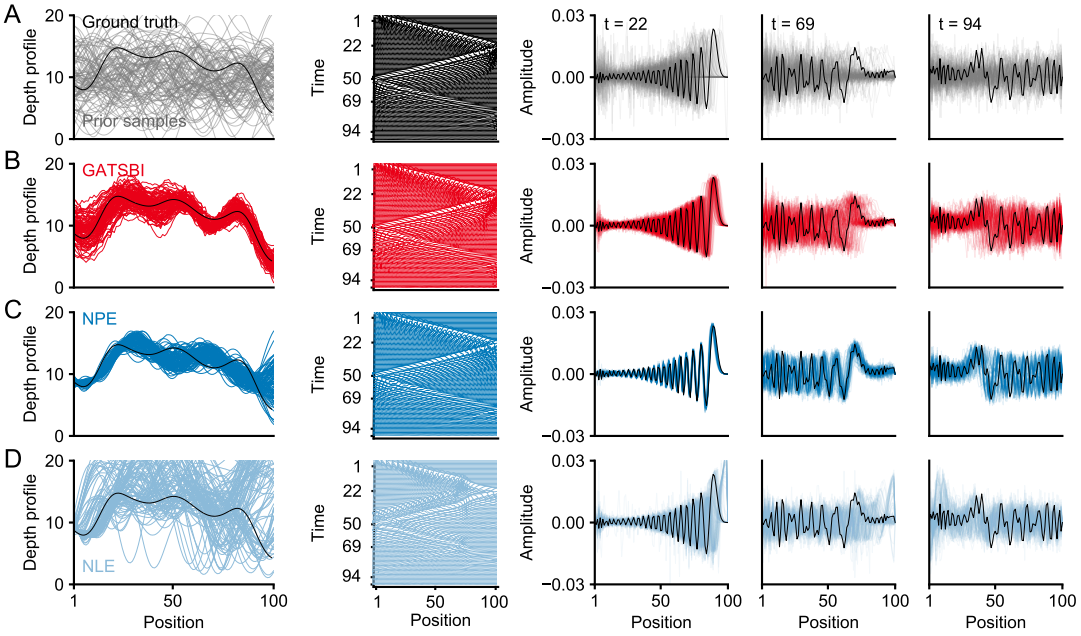

Figure 3: **Shallow water model inference with GATSBI, NPE and NLE.** A. Ground truth, observation and prior samples. Left: ground-truth depth profile and prior samples. Middle: surface wave simulated from ground-truth profile as a function of position and time. Right: wave amplitudes at three different fixed times for ground-truth depth profile (black), and waves simulated from multiple prior samples (gray). B. GATSBI inference. Left: posterior samples (red) versus ground-truth depth profile (black). Middle: surface wave simulated from a single GATSBI posterior sample. Right: wave amplitudes for multiple GATSBI posterior samples, at three different times (red). Ground-truth waves in black. C. NPE inference. Panels as in B. D. NLE inference.

3.2 High-dimensional inference problems

We showed above that, on low-dimensional examples, GATSBI does not perform as well as the best SBI methods. However, we expected that it would excel on problems on which GANs excel: high-dimensional and structured parameter spaces, in which suitably designed neural networks can be used as powerful classifiers. We apply GATSBI to two problems with $\sim$100-dimensional parameter spaces, one of which also has an implicit prior.

Shallow Water Model The 1D shallow water equations describe the propagation of an initial disturbance across the surface of a shallow basin with an irregular depth profile and bottom friction (Figure 3A). They are relevant to oceanographers studying the dynamics on the continental sea shelf (Backhaus, 1983; Holton, 1973). We discretised the partial differential equations on a 100-point spatial grid, obtaining a simulator parameterised by the depth profile of the basin, $\theta \in \mathbb{R}^{100}$ (substantially higher than most previous SBI-applications, which generally are ≈ 10 dim). The resulting amplitude of the waves evolving over a total of 100 time steps constitutes the 10k-dimensional raw observation from which we aim to infer the depth profile. The observation is taken into the Fourier domain, where both the real and imaginary parts receive additive noise and are concatenated, entering the inference pipeline as an array $x \in \mathbb{R}^{20k}$. Our task was to estimate $p(\theta|x)$, i.e., to infer the basin depth profile from a noisy Fourier transform of the surfaces waves. We trained GATSBI with 100k depth profiles sampled from the prior and the resulting surface wave simulations. For comparison, we trained NPE and NLE on the same data, using the same embedding net as the GATSBI discriminator to reduce the high dimensionality of the observations $x \in \mathbb{R}^{20k}$ to $\mathbb{R}^{100}$.

Samples from the GATSBI posterior captured the structure of the underlying ground-truth depth profile (Figure 3B, left). In comparison, only NPE, but not NLE, captured the true shape of the depth profile (Figure 3C, D, left). In addition, the inferred posterior means for GATSBI and NPE, but not NLE, were correlated with the ground-truth parameters (0.88 ± 0.10 for GATSBI and 0.91 ± 0.09 for NPE, mean $\pm$ standard deviation across 1000 different observations). Furthermore, while GATSBI posterior predictive samples did not follow the ground-truth observations as closely as NPE's, they captured the structure and were not as phase-shifted as NLE's (Figure 3 B-D). This is expected since wave-propagation speed depends strongly on depth profile, and thus incorrect inference of the depth parameters leads to temporal shifts between observed and inferred waves. Finally, simulation-based calibration (SBC) (Talts et al., 2020) shows that both NPE and GATSBI provide well-calibrated posteriors on this challenging, 100 dimensional inference problem (clearly, neither is perfectly calibrated, Figure 4). This realistic simulator example illustrates that GATSBI can learn posteriors with a dimensionality far from feasible for most SBI algorithms. It performs similar to NPE, previously reported to be a powerful approach on high-dimensional SBI problems (Lueckmann et al., 2021).

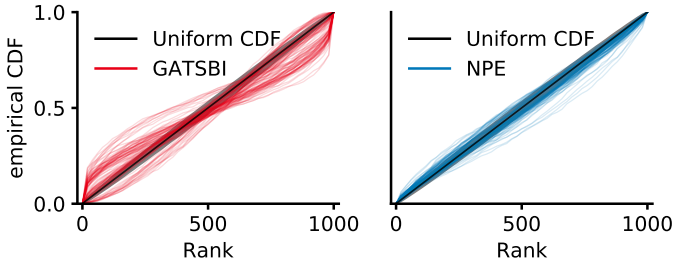

Figure 4: SBC results on shallow water model. Empirical cdf of SBC ranks, one line per posterior dimension (red, GATSBI; blue, NPE). In gray, region showing 99% of deviation expected under the ideal uniform cdf (black).

Noisy Camera Model Finally, we demonstrate a further advantage of GATSBI over many SBI algorithms: GATSBI (like LFVI) can be applied in cases in which the prior distribution is only specified through an implicit model. We illustrate this by considering inference over images from a noisy and blurred camera model. We interpret the model as a black-box simulator. Thus, the clean images θ are samples from an implicit prior over images $\pi(\theta)$, and the simulated sensor activations are observations x. We can then recast the task of denoising the images as an inference problem (we note that our goal is *not* to advance the state-of-the-art for super-resolution algorithms, for which multiple powerful approaches are available, e.g. Kim et al., 2016; Zhang et al., 2018).

Since images are typically high-dimensional objects, learning a posterior over images is already a difficult problem for many SBI algorithms – either due to the implicit prior or the curse of dimensionality making MCMC sampling impracticable. However, the implicit prior poses no challenge for GATSBI since it is trained only with samples.

We chose the EMNIST dataset (Cohen et al., 2017) with 800k 28×28-dimensional images as the implicit prior. This results in an inference problem in a 784-dimensional parameter space—much higher dimensionality than in typical SBI applications. The GATSBI generator was trained to produce clean images, given only the blurred image as input, while the discriminator was trained with clean images from the prior and the generator, and the corresponding blurred images. GATSBI indeed recovered crisp sources from blurred observations: samples from the GATSBI posterior given blurred observations accurately matched the underlying clean ground-truth images (see Figure 5). In contrast,

Figure 5: **Camera model inference.** Top: ground-truth parameter samples from the implicit prior and corresponding blurred camera model observations. Bottom: mean and standard deviation (SD) of inferred GATSBI and NPE posterior samples for matching observation from top.

NPE did not produce coherent posterior samples, and took substantially longer to train than GATSBI (NLE and NRE are not applicable due to the implicit prior).

4 Discussion

We proposed a new method for simulation-based inference in stochastic models with intractable likelihoods. We used conditional GANs to fit a neural network functioning as an implicit density, allowing efficient, scalable sampling. We clarified the relationship between GANs and SBI, and showed how our method is related to adversarial VAEs that learn implicit posteriors over latent variables. We showed how GATSBI can be extended for sequential estimation, fine-tuning inference to a particular observation at the expense of amortisation. In contrast to conventional density-estimation based SBI-methods, GATSBI targets the *reverse* D_{KL}—by thus extending both amortised and sequential SBI, we made it possible to explore the advantages of the forward and reverse D_{KL}.

We found GATSBI to work adequately on two low-dimensional benchmark problems, though it did not outperform neural network-based methods. However, its real potential comes from the ability of GANs to learn generative models in high-dimensional problems. We demonstrated this by using GATSBI to infer a well-calibrated posterior over a 100-dimensional parameter space for the shallow water model, and a 784-dimensional parameter space for the camera model. This is far beyond what is typically attempted with SBI: indeed we found that GATSBI outperformed NPE on the camera model, and performed similarly to NPE and substantially outperformed NLE on the shallow water model (both SBI approaches were previously shown to be more powerful than others in high dimensions; Lueckmann et al., 2021). Moreover, we showed that GATSBI can learn posteriors in a model in which the prior is only defined implicitly (i.e., through a database of samples, not a parametric model).

Despite the advantages that GATSBI confers over other SBI algorithms in terms of flexibility and scalability, it comes with computational and algorithmic costs. GANs are notoriously difficult to train—they are highly sensitive to hyperparameters (as demonstrated in the two-moons example) and problems such as mode collapse are common. Moreover, training time is significantly higher than in other SBI algorithms. In addition, since GATSBI encodes an implicit posterior, i.e., a density from which we can sample but cannot evaluate, it yields posterior samples but not a parametric model. Thus, GATSBI is unlikely to be the method of choice for low-dimensional problems. Similarly, sequential GATSBI requires extensive hyperparameter tuning in order to produce improvements over amortised GATSBI, with the additional cost of training a classifier to approximate the correction factor. Hence, sequential GATSBI might be useful in applications where the computational cost of training is offset by having an expensive simulator. We note that GATSBI might benefit from recent improvements on GAN training speed and sample efficiency (Sauer et al., 2021).

Overall, we expect that GATSBI will be particularly impactful on problems in which the parameter space is high-dimensional and has GAN-friendly structure, e.g. images. Spatially structured models and data abound in the sciences, from climate modelling to ecology and economics. High-dimensional parameter regimes are common, but inaccessible for current SBI algorithms. Building on the strengths of GANs, we expect that GATSBI will help close this gap and open up new application-domains for SBI, and new directions for building SBI methods employing adversarial training.

ACKNOWLEDGMENTS

We thank Kai Logemann for providing code for the shallow water model. We thank Michael Deistler, Marcel Nonnenmacher and Giacomo Bassetto for in-depth discussions; Auguste Schulz, Richard Gao, Artur Speiser and Janne Lappalainen for feedback on the manuscript and the reviewers at ICML 2021, NeurIPS 2021 and ICLR 2022 for insightful feedback. This work was funded by the German Research Foundation (DFG; Germany's Excellence Strategy MLCoE – EXC number 2064/1 PN 390727645; SPP 2041; SFB 1089 'Synaptic Microcircuits'), the German Federal Ministry of Education and Research (BMBF; project ADIMEM, FKZ 01IS18052 A-D; Tübingen AI Center, FKZ: 01IS18039A) and the Helmholtz AI initiative.

ETHICS STATEMENT

Since this is a purely exploratory and theoretical work based on simulated data, we do not foresee major ethical concerns. However, we also note that highly realistic GANs have a potential of being exploited for applications with negative societal impacts e.g. deep fakes (Tolosana et al., 2020).

REPRODUCIBILITY STATEMENT

To facilitate reproducibility, we provide detailed information about simulated data, model set up, training details and experimental results in the Appendix. Code implementing the method and experiments described in the manuscript is available at `https://github.com/mackelab/gatsbi`.

REFERENCES

Jonas Adler and Ozan Öktem. Deep bayesian inversion, 2018.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 214–223. PMLR, 06–11 Aug 2017.

Samaneh Azadi, Catherine Olsson, Trevor Darrell, Ian Goodfellow, and Augustus Odena. Discriminator Rejection Sampling. *arXiv:1810.06758 [cs, stat]*, 2019. arXiv: 1810.06758.

Jan O Backhaus. A semi-implicit scheme for the shallow water equations for application to shelf sea modelling. *Continental Shelf Research*, 2(4):243–254, 1983.

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

Mark A Beaumont, Jean-Marie Cornuet, Jean-Michel Marin, and Christian P Robert. Adaptive approximate Bayesian computation. *Biometrika*, 96(4):983–990, 2009.

Oliver J Britton, Alfonso Bueno-Orovio, Karel Van Ammel, Hua Rong Lu, Rob Towart, David J Gallacher, and Blanca Rodriguez. Experimentally calibrated population of models predicts and explains intersubject variability in cardiac cellular electrophysiology. *Proceedings of the National Academy of Sciences*, 110(23):E2098–E2105, 2013.

Tong Che, Ruixiang Zhang, Jascha Sohl-Dickstein, Hugo Larochelle, Liam Paull, Yuan Cao, and Yoshua Bengio. Your GAN is Secretly an Energy-based Model and You Should use Discriminator Driven Latent Sampling. *arXiv:2003.06060 [cs, stat]*, 2020.

Liqun Chen, Shuyang Dai, Yunchen Pu, Erjin Zhou, Chunyuan Li, Qinliang Su, Changyou Chen, and Lawrence Carin. Symmetric variational autoencoder and connections to adversarial learning. In *International Conference on Artificial Intelligence and Statistics*, pages 661–669. PMLR, 2018.

Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. EMNIST: an extension of MNIST to handwritten letters, 2017.

Alex Cole, Benjamin Kurt Miller, Samuel J. Witte, Maxwell X. Cai, Meiert W. Grootes, Francesco Nattino, and Christoph Weniger. Fast and credible likelihood-free cosmology with truncated marginal neural ratio estimation, 2021.

Kyle Cranmer, Juan Pavez, and Gilles Louppe. Approximating likelihood ratios with calibrated discriminative classifiers. *arXiv preprint arXiv:1506.02169*, 2015.

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.

Conor Durkan, Iain Murray, and George Papamakarios. On contrastive learning for likelihood-free inference. In *Proceedings of the 36th International Conference on Machine Learning*, volume 98 of *Proceedings of Machine Learning Research*. PMLR, 2020.

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, David S. Greenberg, and Jakob H. Macke. Training deep neural density estimators to identify mechanistic models of neural dynamics. *eLife*, 2020.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

David Greenberg, Marcel Nonnenmacher, and Jakob Macke. Automatic posterior transformation for likelihood-free inference. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2404–2414. PMLR, 2019.

Michael U. Gutmann and Jukka Corander. Bayesian optimization for likelihood-free inference of simulator-based statistical models, 2015.

Michael U. Gutmann, Ritabrata Dutta, Samuel Kaski, and Jukka Corander. Likelihood-free inference via classification. *Statistics and Computing*, 28(2):411–425, 2018.

Joeri Hermans, Volodimir Begy, and Gilles Louppe. Likelihood-free MCMC with approximate likelihood ratios. In *Proceedings of the 37th International Conference on Machine Learning*, volume 98 of *Proceedings of Machine Learning Research*. PMLR, 2020.

James R Holton. An introduction to dynamic meteorology. *American Journal of Physics*, 41(5): 752–754, 1973.

Zhiting Hu, Zichao Yang, Ruslan Salakhutdinov, and Eric P Xing. On unifying deep generative models. *arXiv preprint arXiv:1706.00550*, 2017.

Ferenc Huszár. Variational inference using implicit distributions. *arXiv preprint arXiv:1702.08235*, 2017.

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *arXiv preprint arXiv:1611.07004*, 2016.

Vinay Jethava and Devdatt Dubhashi. Easy high-dimensional likelihood-free inference. *arXiv preprint arXiv:1711.11139*, 2017.

Dongjun Kim, Weonyoung Joo, Seungjae Shin, and Il-Chul Moon. Adversarial likelihood-free inference on black-box generator. *arXiv preprint arXiv:2004.05803*, 2020.

J. Kim, J. K. Lee, and K. M. Lee. Accurate image super-resolution using very deep convolutional networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1646–1654, 2016.

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

Athanasios Kousathanas, Christoph Leuenberger, Jonas Helfer, Mathieu Quinodoz, Matthieu Foll, and Daniel Wegmann. Likelihood-free inference in high-dimensional models, 2015.

Brodie AJ 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*, 4(1):e1701676, 2018.

Gilles Louppe, Joeri Hermans, and Kyle Cranmer. Adversarial variational optimization of non-differentiable simulators. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 1438–1447, 2017.

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 30*, pages 1289–1299. Curran Associates, Inc., 2017.

Jan-Matthis Lueckmann, Jan Boelts, David Greenberg, Pedro Goncalves, and Jakob Macke. Benchmarking simulation-based inference. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, pages 343–351, 2021.

Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

P Marjoram, J Molitor, V Plagnol, and S Tavare. Markov chain Monte Carlo without likelihoods. *Proceedings of the National Academy of Sciences*, 100(26), 2003.

Edward Meeds and Max Welling. Gps-abc: Gaussian process surrogate approximate bayesian computation, 2014.

Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. Adversarial variational Bayes: Unifying variational autoencoders and generative adversarial networks. *34th International Conference on Machine Learning, ICML 2017*, 5:3694–3707, 2017.

Cheol-Hui Min, Jinseok Bae, Junho Lee, and Young Min Kim. Gatsbi: Generative agent-centric spatio-temporal object interaction, 2021.

Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

Shakir Mohamed and Balaji Lakshminarayanan. Learning in implicit generative models. *arXiv preprint arXiv:1610.03483*, 2016.

Fabio Muratore, Fabio Ramos, Greg Turk, Wenhao Yu, Michael Gienger, and Jan Peters. Robot learning from randomized simulations: A review. *arXiv preprint arXiv:2111.00956*, 2021.

Victor M.-H. Ong, David J. Nott, Minh-Ngoc Tran, Scott A. Sisson, and Christopher C. Drovandi. Likelihood-free inference in high dimensions with synthetic likelihood. *Computational Statistics & Data Analysis*, 128:271–291, 2018. ISSN 0167-9473.

George Papamakarios and Iain Murray. Fast ϵ-free inference of simulation models with Bayesian conditional density estimation. In *Advances in Neural Information Processing Systems 29*, pages 1028–1036. Curran Associates, Inc., 2016.

George Papamakarios, David Sterratt, and Iain Murray. Sequential neural likelihood: Fast likelihood-free inference with autoregressive flows. In *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 89 of *Proceedings of Machine Learning Research*, pages 837–848. PMLR, 2019.

George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64, 2021.

Jaimit Parikh, James Kozloski, and Viatcheslav Gurev. Integration of AI and mechanistic modeling in generative adversarial networks for stochastic inverse problems. *arXiv preprint arXiv:2009.08267*, 2020.

Kim Cuc Pham, David J Nott, and Sanjay Chaudhuri. A note on approximating ABC-MCMC using flexible classifiers. *STAT*, 3(1):218–227, 2014.

Stefan T Radev, Ulf K Mertens, Andreas Voss, Lynton Ardizzone, and Ullrich Köthe. Bayesflow: Learning complex stochastic models with invertible neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

CE. Rasmussen and CKI. Williams. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. MIT Press, 2006.

G. S. Rodrigues, D. J. Nott, and S. A. Sisson. Likelihood-free approximate gibbs sampling, 2019.

Axel Sauer, Kashyap Chitta, Jens Müller, and Andreas Geiger. Projected gans converge faster. *Bridging Theory and Practice Workshop at Neural Information Processing Systems*, 2021.

Scott A Sisson, Yanan Fan, and Mark M Tanaka. Sequential Monte Carlo without likelihoods. *Proceedings of the National Academy of Sciences*, 104(6):1760–1765, 2007.

Akash Srivastava, Lazar Valkov, Chris Russell, Michael U Gutmann, and Charles Sutton. Veegan: Reducing mode collapse in gans using implicit variational learning. *arXiv preprint arXiv:1705.07761*, 2017.

Sean Talts, Michael Betancourt, Daniel Simpson, and Aki Vehtari. Validating Bayesian Inference Algorithms with Simulation-Based Calibration. pages 1–26, 2020.

Simon Tavaré, David J. Balding, R. C. Griffiths, and Peter Donnelly. Inferring coalescence times from DNA sequence data. *Genetics*, 145(2), 1997.

Owen Thomas, Ritabrata Dutta, Jukka Corander, Samuel Kaski, and Michael U. Gutmann. Likelihood-Free Inference by Ratio Estimation. *Bayesian Analysis*, pages 1–31, 2021.

Ruben Tolosana, Ruben Vera-Rodriguez, Julian Fierrez, Aythami Morales, and Javier Ortega-Garcia. Deepfakes and beyond: A survey of face manipulation and fake detection, 2020.

Dustin Tran, Rajesh Ranganath, and David Blei. Hierarchical implicit models and likelihood-free variational inference. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Richard Eric Turner and Maneesh Sahani. *Two problems with variational expectation maximisation for time series models*, page 104–124. Cambridge University Press, 2011.

Kevin Yu, Harnaik Dhami, Kartik Madhira, and Pratap Tokekar. Gatsbi: An online gtsp-based algorithm for targeted surface bridge inspection, 2020.

Manzil Zaheer, Chun-Liang Li, Barnabás Póczos, and Ruslan Salakhutdinov. GAN connoisseur: Can GANs learn simple 1d parametric distributions? *Bridging Theory and Practice Workshop at Neural Information Processing Systems*, 2017.

Yulun Zhang, Kunpeng Li, Kai Li, Lichen Wang, Bineng Zhong, and Yun Fu. Image super-resolution using very deep residual channel attention networks. In *Computer Vision – ECCV 2018*, pages 294–310. Springer International Publishing, 2018.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv preprint arXiv:1703.10593*, 2017.

APPENDIX

A PROOFS

A.1 PROOF OF CONVERGENCE FOR GATSBI

Proposition 1. *Given a fixed generator f_ϕ, the discriminator $\mathrm{D}_{\psi^*}(\theta, x)$ maximising equation 2 satisfies*

$$\mathrm{D}_{\psi^*}(\theta, x) = \frac{p(\theta|x)}{p(\theta|x) + q_\phi(\theta|x)},$$

and the corresponding loss function for the generator parameters is the Jensen-Shannon divergence (JSD) between the true and approximate posterior:

$$L_{\psi^*}(\phi) = 2\,\mathrm{JSD}(p(\theta|x) \;||\; q_\phi(\theta|x)) - \log\ 4$$

Proof. We start with equation 2. The proof proceeds as for Proposition 1 and 2 in Goodfellow et al. (2014). For convenience, we elide the arguments of the various quantities, so that q_ϕ denotes $q_\phi(\theta|x)$, p denotes $p(\theta|x)$ and D_ψ denotes $\mathrm{D}_\psi(\theta, x)$.

$$\begin{aligned}
L(\phi, \psi) &= \mathbb{E}_{p(x)}\Big[\mathbb{E}_p \log \mathrm{D}_\psi + \mathbb{E}_{q_\phi} \log\big(1 - \mathrm{D}_\psi\big)\Big] \\
&= \mathbb{E}_{p(x)}\left[\int p \log \mathrm{D}_\psi \ \mathrm{d}\theta + \int q_\phi \log(1 - \mathrm{D}_\psi)\ \mathrm{d}\theta\right] \\
&= \mathbb{E}_{p(x)}\Big[\int \Big(p \log \mathrm{D}_\psi + q_\phi \log(1 - \mathrm{D}_\psi)\Big)\ \mathrm{d}\theta\Big].
\end{aligned}$$

For any function $g(x) = a \log x + b \log(1-x)$, where $a, b \in \mathbb{R}^2 \setminus \{0, 0\}$ and $x \in (0, 1)$, the maximum of the function is at $x = {}^{a}\!/_{a+b}$. Hence, $L(\phi, \psi)$, for a fixed ϕ, achieves it's maximum at:

$$\mathrm{D}_{\psi^*} = \frac{p}{p + q_\phi}$$

Note that $p(x)$ drops out of the expression for D_{ψ^*} since it is common to both terms in $L(\phi, \psi)$. Plugging this into equation 2 and dropping the expectation over x without loss of generality:

$$\begin{aligned}
L_{\psi^*}(\phi) &= \mathbb{E}_p\ \log \frac{p}{p + q_\phi} + \mathbb{E}_{q_\phi} \log\ \frac{q_\phi}{p + q_\phi} \\
&= \mathbb{E}_p\ \log \frac{2\,p}{p + q_\phi} + \mathbb{E}_{q_\phi} \log \frac{2\,q_\phi}{p + q_\phi} - \log\ 4 \\
&= 2\,\mathrm{JSD}(p \;||\; q_\phi) - \log\ 4.
\end{aligned}$$

The JSD is always non-negative, and zero only when $p(\theta|x) = q_\phi(\theta|x)$. Thus, the global minimum $L_{\psi^*}(\phi^*) = -\log\ 4$ is achieved only when the GAN generator converges to the ground-truth posterior $p(\theta|x)$. □

A.2 PROOF FOR GATSBI MAXIMIZING ELBO LOSS

Proposition 2. *If ϕ^* and ψ^* denote the Nash equilibrium of a min-max game defined by equation 2 then ϕ^* also maximises the evidence lower bound of a VAE with a fixed decoder i.e.,*

$$\phi^* = \arg\max_\phi L_{\mathrm{E}}(\phi) \tag{8}$$

$$= \arg\max_\phi\ \mathbb{E}_{p(x)}\mathbb{E}_{q_\phi(\theta|x)}\left(\log \frac{\pi(\theta)}{q_\phi(\theta|x)} + \log p(x|\theta)\right) \tag{9}$$

Proof. The proposition is trivially true if $q_{\phi^*}(\theta|x) = p(\theta|x)$, i.e., $p(\theta|x)$ is the true minimum of both the JSD and D_{KL}. However, we here show that the equivalence holds even when $q_{\phi^*}(\theta|x)$ is *not*

the true posterior, e.g. if $q_\phi(\theta|x)$ belongs to a family of distributions that does not include the true posterior.

Given that ϕ^* and ψ^* denote the Nash equilibrium in equation 2, we know from Proposition 1 that the optimal discriminator is given by

$$\mathrm{D}_{\psi^*}(\theta, x) = \frac{p(\theta|x)}{p(\theta|x) + q_{\phi^*}(\theta|x)}.$$

To lighten notation, we elide the parameters of the networks and their arguments, denoting $D_{\psi^*}(\theta|x)$ as D_{ψ^*}, $q_{\phi^*}(\theta|x)$ as q^*, and so on. If we plug D_{ψ^*} into Eq. 2, we have

$$\begin{aligned}\phi^* &= \arg\min_\phi \ \mathbb{E}_{p(\theta|x)} \ \log \ \mathrm{D}_{\psi^*} + \mathbb{E}_{q_\phi}\Big(\log\big(1 - \mathrm{D}_{\psi^*}\big)\Big) \\ &= \arg\min_\phi \mathbb{E}_{q_\phi} \ \log \ \big(1 - \mathrm{D}_{\psi^*}\big) \end{aligned} \tag{10}$$

Note that this is true *only* at the Nash equilibrium, where D_{ψ^*} is a function of q_{ϕ^*} and *not* q_ϕ. This allows us to drop the first term from the equation. In other words, if we switch out q_{ϕ^*} with any other q_ϕ in the expectation, equation 10 is no longer minimum w.r.t ϕ, even though D_{ψ^*} is optimal.

Let us define $\mathrm{D}_\psi := \sigma(R_\psi)$, where $\sigma(\cdot) := {}^1\!/_{1 + \exp(-\,\cdot)}$. Then from equation 3,

$$\sigma(R_{\psi^*}) = \frac{p(\theta|x)}{p(\theta|x) + q_{\phi^*}} \implies \frac{1}{1 + \mathrm{e}^{-R_{\psi^*}}} = \frac{1}{1 + {}^{q_{\phi^*}}\!/_{p(\theta|x)}}$$

Comparing the l.h.s. and r.h.s. of the equation above, we get

$$R_{\psi^*} = \log \frac{p(\theta|x)}{q_{\phi^*}}. \tag{11}$$

Since both $\log$ and σ are monotonically increasing functions, we also have from equation 10:

$$\begin{aligned}\phi^* &= \arg\min_\phi \mathbb{E}_{q_\phi} \ \log\big(1 - \sigma(R^*)\big) \\ &= \arg\min_\phi \mathbb{E}_{q_\phi} \ \big(1 - \sigma(R_{\psi^*})\big) \\ &= \arg\max_\phi \mathbb{E}_{q_\phi} \ \sigma(R_{\psi^*}) \\ &= \arg\max_\phi \mathbb{E}_{q_\phi}(R_{\psi^*}) \end{aligned} \tag{12}$$

In other words, q_{ϕ^*} maximises the function $\mathbb{E}_{q_\phi}(R_{\psi^*})$. Now, to prove equation 9, we need to show that $L_{\mathrm{E}}(\phi) < L_{\mathrm{E}}(\phi^*) \ \forall \ \ \phi \neq \phi^*$.

$$\begin{aligned}
L_{\mathrm{E}}(\phi) &= \mathbb{E}_{p(x)}\mathbb{E}_{q_\phi}\left(\log \pi(\theta) - \log q_\phi + \log p(x|\theta)\right) \\
&= \mathbb{E}_{p(x)}\mathbb{E}_{q_\phi}\left(\log \frac{p(\theta|x)}{q_\phi} + \log \frac{p(x|\theta)\pi(\theta)}{p(\theta|x)}\right) \\
&= \mathbb{E}_{p(x)}\mathbb{E}_{q_\phi}\left(\log \frac{p(\theta|x)}{q_\phi} + \log p(x)\right) \\
&= \mathbb{E}_{p(x)}\mathbb{E}_{q_\phi}\left(\log \frac{p(\theta|x)}{q_{\phi^*}} + \log p(x)\right) - \mathbb{E}_{p(x)}(D_{\mathrm{KL}}(q_\phi||q_{\phi^*})) \\
&< \ \mathbb{E}_{p(x)}\mathbb{E}_{q_\phi}\left(\log \frac{p(\theta|x)}{q_{\phi^*}} + \log p(x)\right) \\
&= \mathbb{E}_{p(x)}\mathbb{E}_{q_\phi}\left(R_{\psi^*} + \log p(x)\right) && \text{from equation 11} \\
&< \ \mathbb{E}_{p(x)}\mathbb{E}_{q_{\phi^*}}\left(R_{\psi^*}\right) + \mathbb{E}_{p(x)}\mathbb{E}_{q_\phi} \log p(x) && \text{from equation 12} \\
&= \mathbb{E}_{p(x)}\mathbb{E}_{q_{\phi^*}}\left(R_{\psi^*} + \log p(x)\right) + \mathbb{E}_{p(x)}\mathbb{E}_{q_\phi} \log p(x) - \mathbb{E}_{p(x)}\mathbb{E}_{q_{\phi^*}} \log p(x) \\
&= \mathbb{E}_{p(x)}\mathbb{E}_{q_{\phi^*}}\left(\log \frac{p(\theta|x)}{q_{\phi^*}} + \log p(x)\right) \\
&= L_{\mathrm{E}}(\phi^*) \\
\implies L_{\mathrm{E}}(\phi) &< L_{\mathrm{E}}(\phi^*)
\end{aligned}$$

Hence, the approximate posterior obtained by optimising the GAN objective also maximises the evidence lower bound of the corresponding VAE. □

A.3 CONNECTION BETWEEN LFVI, DEEP POSTERIOR SAMPLING AND GATSBI

Adversarial inference approaches maximise the Evidence Lower Bound (ELBO) equation 6 to train a VAE, and use a discriminator to approximate intractable ratios of densities in the loss (see Table 1). Likelihood-free variational inference (Tran et al., 2017, LFVI) is one such method.

Table 1: Comparison of adversarial inference algorithms: BiGAN (Donahue et al., 2019), ALI (Dumoulin et al., 2016), AAE (Makhzani et al., 2015), AVB (Mescheder et al., 2017), and LFVI (Tran et al., 2017).

ALGORITHM	DISCRIMINATOR RATIO	GENERATOR LOSS FUNCTION
BIGAN, ALI	$p_\alpha(x\|u)p(u)/q_\phi(u\|x)p(x)$	$\mathrm{JSD}(p_\alpha(u,x)\|\|q_\phi(u,x))$
AAE	$p(u)/q_\phi(u)$	$\mathrm{JSD}(p(u)\|\|q_\phi(u))$
AVB	$p(u)p(x)/q_\phi(u\|x)p(x)$	$D_{\mathrm{KL}}(q_\phi(u\|x)\|\|p(u))$
LFVI	$p(u\|x)p(x))/q_\phi(u\|x)p(x)$	$D_{\mathrm{KL}}(q_\phi(u\|x)\|\|p(u\|x))$
GATSBI	$p(u\|x)p(x)/q_\phi(u\|x)p(x)$	$\mathrm{JSD}(p(u\|x)\|\|q_\phi(u\|x))$

In the most general formulation, LFVI learns a posterior over both latents u and global parameters β which are latents shared across multiple observations i.e., $q_\phi(z, \beta|x)$ and maximises the ELBO given by

$$L_{\mathrm{LF}}(\phi) = \mathbb{E}_{q_\phi(\beta)} \log \frac{p(\beta)}{q_\phi(\beta)} + \mathbb{E}_{q_\phi(u|x)p'(x)} \log \frac{p(x|u)p(u)}{q_\phi(u|x)p'(x)} + \text{const.} \tag{13}$$

where $p'(x)$,[3] an *empirical distribution* over observations, and *not* necessarily $p(x)$, the marginal likelihood of the simulator. A discriminator, $\mathrm{D}_\psi(x, u)$,[4] is trained with the cross-entropy loss to approximate the intractable ratio $\frac{p(x|u)p(u)}{q_\phi(u|x)p'(x)}$ in the second term. Using the nomenclature from Huszár (2017), we note that D_ψ is *joint-contrastive*: it simultaneously discriminates between tuples $(x, u) \sim p(x|u)p(u)$ and $(\hat{x}, \hat{u}) \sim q_\phi(\hat{u}|\hat{x})p'(\hat{x})$. GATSBI, by contrast, is *prior-contrastive*: its discriminator only discriminates between parameters θ, given a fixed x.

However, when $p'(x) = p(x)$ and β is constant, the LFVI discriminator becomes prior-contrastive, equation 13 is a function of both the discriminator parameters ψ and generator parameters ϕ, and it differs from GATSBI only by a single term i.e., from equation 2 and equation 13 and ignoring the constant:

$$L_{\mathrm{GATSBI}}(\phi, \psi) = -L_{\mathrm{LF}}(\phi, \psi) + \mathbb{E}_{q_\phi(u|x)p(x)} \log \mathrm{D}_\psi(x, u) \tag{14}$$

The second term corresponds to the non-saturating GAN loss (Goodfellow et al., 2014). In this

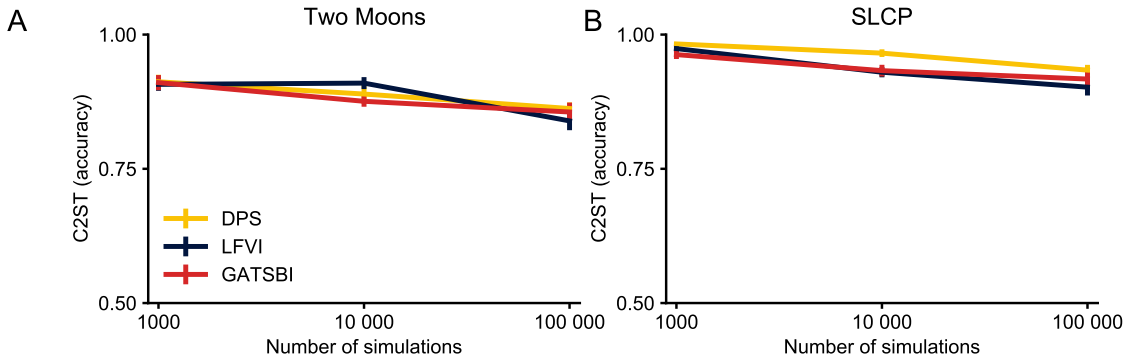

Figure 6: **GATSBI, LFVI and Deep Poseterior Sampling (DPS) on benchmark problems.** Mean C2ST score (± standard error of the mean) for 10 observations each. A. On Two Moons, the C2ST scores for GATSBI (red), LFVI (navy) and Deep Posterior Sampling (DPS, yellow) are qualitatively similar across all simulation budgets B. On SLCP, DPS is slightly worse than LFVI and GATSBI.

[3] In Tran et al. (2017), $p'(x)$ is denoted $q(x)$

[4] denoted $r_\psi(x, u)$ in Tran et al. (2017)

setting, with an optimal discriminator, GATSBI minimises a JSD, whereas LFVI minimises the reverse D_{KL}.

Adler and Öktem (2018) introduce Deep Posterior Sampling which also implements an adversarial algorithm for posterior estimation. The set-up is similar to GATSBI, but GANs are trained using a Wasserstein loss as in Arjovsky et al. (2017). The Wasserstein loss imposes stronger conditions on the GAN networks in order for the generator to recover the target distribution, i.e., the discriminator has to be 1-Lipschitz and the generator K-Lipschitz (Arjovsky et al., 2017; Qi, 2018). However, the JSD loss function allowed us to outline GATSBI's connection to adversarial VAEs and subsequently its advantages for SBI (see Sec. 2.4, Suppl. Sec. A.2 and the discussion above). Whether this would also be possible with the Wasserstein loss remains a subject for future work. Nevertheless, the sequential extension of GATSBI using the energy-based correction (see Sec. 2.5 and Supp. Sec. A.4) could in principle also be used with the Wasserstein metric.

Mescheder et al. (2018) state that the WGAN converges only when the discriminator minimizes the Wasserstein metric at every step, which does not happen in practice. Fedus et al. (2017) argue that a GAN generator does not necessarily minimise a JSD at *every* update step since the discriminator is optimal only in the limit of infinite data. Hence, neither asymptotic property can be used to reason about GAN behaviour in practice. As a consequence, it is difficult to predict the conditions under which LFVI, or Deep Posterior Sampling would outperform GATSBI, or vice-versa. Nevertheless, given the same networks and hyperperparameters (with slight modifications to the discriminator for Deep Posterior Sampling, see App. D.1 for details), we found empirically that LFVI, Deep Posterior Sampling and GATSBI are qualitatively similar on Two Moons, and that Deep Posterior Sampling is slightly worse that the other two algorithms on SLCP: i.e., there is no advantage to using one algorithm over the other on the problems investigated (see Fig. 6).

A.4 SEQUENTIAL GATSBI

For many SBI applications, the density estimator is only required to generate good posterior samples for a particular experimentally observed data x_{o}. This can be achieved by training the density estimator using samples θ from a proposal prior $\tilde{\pi}(\theta)$ instead of the prior $\pi(\theta)$. The proposal prior ideally produces parameters θ that are localised around the modes of $p(\theta|x_{\mathrm{o}})$ and can guide the density estimator towards inferring a posterior that is accurate for $x \approx x_{\mathrm{o}}$. If we replace the true prior $\pi(\theta)$ with a proposal prior $\tilde{\pi}(\theta) = q_\phi(\theta|x_{\mathrm{o}})$, i.e., the posterior estimated by GATSBI, and sample from the respective distributions

$$\theta, x \sim \tilde{\pi}(\theta)p(x|\theta),$$

the corresponding GAN loss (from equation equation 2) is:

$$\tilde{L}(\phi, \psi) = \mathbb{E}_{\tilde{\pi}(\theta)p(x|\theta)p(z)} \left[\log \mathrm{D}_\psi(\theta, x) + \log(1 - \mathrm{D}_\psi(f_\phi(z, x), x))\right] \tag{15}$$

$$= \mathbb{E}_{\tilde{p}(\theta|x)\tilde{p}(x)} \log \mathrm{D}_\psi(\theta, x) + \mathbb{E}_{q_\phi(\theta|x)\tilde{p}(x)} \log(1 - \mathrm{D}_\psi(\theta, x)). \tag{16}$$

This loss would allow us to obtain a generator that produces samples θ that are likely to generate outputs x close to x_o, when plugged into the simulator. However, Proposition 1 in Appendix A.1 shows that this loss would result in $q_\phi(\theta|x)$ converging to the proposal posterior $\tilde{p}(\theta|x)$ rather than the ground-truth posterior $p(\theta|x)$. In order to learn a conditional density that is accurate for $x \approx x_{\mathrm{o}}$ but nevertheless converges to the correct posterior, we need to correct the approximate posterior for the bias due to the proposal prior. We outline three different approaches to this correction step:

Using energy-based GANs Although it is possible to use correction factors directly in the GATSBI loss function, as we outline in the next section, these corrections can lead to unstable training (Papamakarios et al., 2019). Here, we outline an approach in which we train on samples from the proposal prior without explicitly introducing correction factors into the loss function. Instead, we change the setup of the generator to produce 'corrected' samples, which are then used to compute the usual cross-entropy loss, and finally we train the discriminator and generator.

Let us start by introducing the correction factor $\omega(\theta, x) = \frac{\pi(\theta)}{\tilde{\pi}(\theta)} \frac{\tilde{p}(x)}{p(x)}$, such that

$$p(\theta|x) = \tilde{p}(\theta|x)\ \omega(\theta, x). \tag{17}$$

In the original formulation of GATSBI, sampling from $q_\phi(\theta|x)$ entails sampling latents $z \sim p(z)$, and transforming them by a deterministic function $f_\phi(x, z)$ to get parameters θ (see equation equation 1).

Following the energy-based GAN formulation (Azadi et al., 2019; Che et al., 2020), we define an *intermediate* latent distribution $p_t(z)$:

$$p_t(z) = p(z)(\omega(f_\phi(x, z), x))^{-1}. \tag{18}$$

$p_t(z)$ is the distribution of latent variables that, when passed through the function $f_\phi(x, z)$, are most likely to produce samples from the *approximate proposal posterior* $\tilde{q}_\phi(\theta|x) = q_\phi(\theta|x)(\omega(\theta, x))^{-1}$. For GANs, $p(z)$ is typically a tractable distribution whose likelihood can be computed, and from which one can sample, and thus, we can use MCMC or rejection sampling to also sample from $p_t(z)$ (see Appendix D.1 for details). The resulting loss function is:

$$\tilde{L}(\phi, \psi) = \mathbb{E}_{\tilde{p}(\theta|x)\tilde{p}(x)} \log \mathrm{D}_\psi(\theta, x) + \mathbb{E}_{p_t(z)\tilde{p}(x)} \log(1 - \mathrm{D}_\psi(f_\phi(x, z), x)) = \mathbb{E}_{\tilde{p}(x)}\big[L_1 + L_2\big]. \tag{19}$$

Note that there are no explicit correction factors in the loss.

We now show that optimising the loss function equation 19 leads to the generator converging to the correct posterior distribution. Let us first focus on the second term L_2:

$$\begin{aligned}
L_2 &= \mathbb{E}_{p_t(z)} \log(1 - \mathrm{D}_\psi(f_\phi(x, z), x)) \\
&= \int p_t(z) \log(1 - \mathrm{D}_\psi(f_\phi(x, z), x)) \\
&= \int p(z)\ (\omega(f_\phi(x, z), x))^{-1}\ \log(1 - \mathrm{D}_\psi(f_\phi(x, z), x)) && \text{from equation 18} \\
&= \int q_\phi(\theta|x)\ (\omega(\theta, x))^{-1}\ \log(1 - \mathrm{D}_\psi(\theta, x)) && \text{reparam. trick} \\
&= \int \tilde{q}_\phi(\theta|x)\ \log(1 - \mathrm{D}_\psi(\theta, x)) \\
&= \mathbb{E}_{\tilde{q}_\phi(\theta|x)} \log(1 - \mathrm{D}_\psi(\theta, x)).
\end{aligned}$$

Thus, from Proposition 1, we can conclude that by optimising the loss function equation 19, $\tilde{q}_\phi(\theta|x) \to \tilde{p}(\theta|x)$. This implies that the generator network, which represents $q_\phi(\theta|x)$, converges to $p(\theta|x)$, since both the approximate and target proposal posteriors are related respectively to the approximate and true posteriors by the same multiplicative factor $\omega(\theta, x)$. Note that the generator is more accurate in estimating the posterior given x_o (or $x \approx x_\mathrm{o}$), i.e., $p(\theta|x_\mathrm{o})$ than given x far from x_o, since it is trained on samples from the proposal prior.

In practice, this scheme does produce improvements in the learned posterior. However, it is computationally expensive, because *every update* to the generator and discriminator requires a round of MCMC or rejection sampling to obtain the 'corrected' samples. Moreover, if we use the generator from the previous round as the proposal prior in the next round, we need to train a classifier to approximate $\omega(\theta, x)$ at every round.[5] Finally, this approach has additional hyperparameters that need to be tuned during GAN training, which could make it prohibitively difficult to use for most applications.

Below, we outline theoretical arguments for two additional approaches, although we only provide empirical results for the second approach.

Using importance weights Lueckmann et al. (2017) solve the problem of bias from using a proposal prior by introducing importance weights in their loss function. One can use the same trick for GATSBI,by introducing the importance weights $\omega(\theta, x) = \frac{\pi(\theta)}{\tilde{\pi}(\theta)} \frac{\tilde{p}(x)}{p(x)}$ into the loss defined in equation 16:

$$\tilde{L}(\phi, \psi) = \mathbb{E}_{\tilde{\pi}(\theta)p(x|\theta)p(z)} \left[\omega(\theta, x) \log \mathrm{D}_\psi(\theta, x) + \log(1 - \mathrm{D}_\psi(f_\phi(z, x), x))\right] = L_1 + L_2. \tag{20}$$

[5]Note that the correction factor could be computed in closed form if we had a generator with an evaluable density: we would not have to train a classifier to approximate it.

Optimising this loss allows $q_\phi(\theta|x)$ to converge to the true posterior $p(\theta|x)$. Let us focus on the first term L_1:

$$\begin{aligned}
L_1 &= \mathbb{E}_{\tilde{p}(\theta|x)\tilde{p}(x)}\omega(\theta,x)\log \mathrm{D}_\psi(\theta,x) \\
&= \int_x\int_\theta \tilde{p}(x)\tilde{p}(\theta|x)\frac{\pi(\theta)}{\tilde{\pi}(\theta)}\frac{\tilde{p}(x)}{p(x)}\log \mathrm{D}_\psi(\theta,x) \\
&= \int_x\int_\theta p(x|\theta)\tilde{\pi}(\theta)\frac{\pi(\theta)}{\tilde{\pi}(\theta)}\frac{\tilde{p}(x)}{p(x)}\log \mathrm{D}_\psi(\theta,x) \\
&= \int_x\int_\theta p(x|\theta)\pi(\theta)\frac{\tilde{p}(x)}{p(x)}\log \mathrm{D}_\psi(\theta,x) \\
&= \int_x\int_\theta p(x)p(\theta|x)\frac{\tilde{p}(x)}{p(x)}\log \mathrm{D}_\psi(\theta,x) \\
&= \int_x\int_\theta \tilde{p}(x)p(\theta|x)\log \mathrm{D}_\psi(\theta,x) \\
&= \mathbb{E}_{\tilde{p}(x)p(\theta|x)}\log \mathrm{D}_\psi(\theta,x).
\end{aligned}$$

Thus, from Proposition 1, we can conclude that by optimising the loss function equation 20, the generator $q_\phi(\theta|x)$ converges to the true posterior. However, the importance-weight correction could lead to high-variance gradients (Papamakarios et al., 2019). This would be particularly problematic for GANs, where the loss landscape for each network is modified with updates to its adversary, and there is no well-defined optimum. High-variance gradients could cause training to take longer or even prevent it from converging altogether.

Using inverse importance weights Since using importance weights in the loss can lead to high-variance gradients, we could instead consider using the inverse of the importance weights $(\omega(\theta,x))^{-1} = \frac{\tilde{\pi}(\theta)}{\pi(\theta)}\frac{p(x)}{\tilde{p}(x)}$ in the second term in equation 16:

$$\tilde{L}(\phi,\psi) = \mathbb{E}_{\tilde{p}(\theta|x)\tilde{p}(x)}\log \mathrm{D}_\psi(\theta,x) + \mathbb{E}_{q_\phi(\theta|x)\tilde{p}(x)}\ (\omega(\theta,x))^{-1}\ \log(1-\mathrm{D}_\psi(\theta,x)) = L_1 + L_2. \tag{21}$$

Optimising $\tilde{L}(\phi,\psi)$ from equation 21 will result in the generator approximating the true posterior at convergence. Focusing on the second term of the loss function L_2:

$$\begin{aligned}
L_2 &= \mathbb{E}_{q_\phi(\theta|x)\tilde{p}(x)}\ (\omega(\theta,x))^{-1}\ \log(1-\mathrm{D}_\psi(\theta,x)) \\
&= \iint \tilde{p}(x)q_\phi(\theta|x)\frac{\tilde{\pi}(\theta)}{\pi(\theta)}\frac{p(x)}{\tilde{p}(x)}\log(1-\mathrm{D}_\psi(\theta,x)) \\
&= \iint \tilde{p}(x)\tilde{q}_\phi(\theta|x)\log(1-\mathrm{D}_\psi(\theta,x)) \\
&= \mathbb{E}_{\tilde{p}(x)\tilde{q}_\phi(\theta|x)}\log(1-\mathrm{D}_\psi(\theta,x)).
\end{aligned}$$

Thus, from Proposition 1, we can conclude that by optimising the loss function equation 21, $\tilde{q}_\phi(\theta|x)$ converges to $\tilde{p}(\theta|x)$. Since $\tilde{q}_\phi(\theta|x)$ differs from $q_\phi(\theta|x)$ by the same factor as $\tilde{p}(\theta|x)$ from $p(\theta|x)$, i.e., $(\omega(\theta,x))^{-1}$ (see equation equation 17), this implies that $q_\phi(\theta|x) \to p(\theta|x)$.

B TRAINING ALGORITHMS

Algorithm 1 GATSBI

Input : prior $\pi(\theta)$, simulator $p(x|\theta)$, generator f_ϕ, discriminator D_ψ, learning rate λ
Output: Trained GAN networks f_{ϕ^*} and D_{ψ^*}
$\Theta = \{\theta_1, \theta_2, \ldots, \theta_n\} \overset{\text{i.i.d}}{\sim} \pi(\theta)$
$\mathbf{X} = \{x_1, x_2, \ldots, x_n\} \sim p(x_i|\theta_i)$
while not converged **do**
  for discriminator iterations **do**
    Sample mini-batch $\mathbf{X}_d$, Θ_d from $\mathbf{X}$, Θ
    $\mathbf{Z} \sim p(z)$, $\hat{\Theta}_d = f_\phi(\mathbf{Z}, \mathbf{X}_d)$
    $L = \sum_{\mathbf{X}_d} \left(\sum_{\Theta_d} \log \mathrm{D}_\psi(\Theta_d, \mathbf{X}_d) + \sum_{\hat{\Theta}_d} \log(1 - \mathrm{D}_\psi(\hat{\Theta}_d, \mathbf{X}_d)) \right)$
    $\psi \leftarrow \psi + \lambda \nabla_\psi L$
  end for
  for generator iterations **do**
    Sample mini-batch $\mathbf{X}_g$, Θ_g from $\mathbf{X}$, Θ
    $\mathbf{Z} \sim p(z)$, $\hat{\Theta}_g = f_\phi(\mathbf{Z}, \mathbf{X}_g)$
    $L = -\sum_{\mathbf{X}_g} \sum_{\hat{\Theta}_g} \log(1 - \mathrm{D}_\psi(\hat{\Theta}_g, \mathbf{X}_g))$
    $\phi \leftarrow \phi + \lambda \nabla_\phi L$
  end for
end while

Algorithm 2 Sequential GATSBI with energy-based correction

Input: $\pi(\theta)$, simulator $p(x|\theta)$, classifier ω = 1, observation x_o, f_ϕ, D_ψ, learning rate λ
Output: Trained GAN networks f_{ϕ^*} and D_{ψ^*}
for $i = 1 \cdots$ number of rounds **do**
  $\Theta_i = \{\theta_1, \theta_2, \ldots, \theta_n\} \overset{\text{i.i.d}}{\sim} \pi(\theta)$
  $\mathbf{X}_i = \{x_1, x_2, \ldots, x_n\} \sim p(x|\theta)$
  if $i > 1$ **then**:
    $\omega_\theta \leftarrow \max_{\omega_\theta} \log \sigma(\omega_\theta(\Theta_0)) + \log(1 - \sigma(\omega_\theta(\Theta_i)))$
    $\omega_x \leftarrow \max_{\omega_x} \log \sigma(\omega_x(\mathbf{X}_0)) + \log(1 - \sigma(\omega_x(\mathbf{X}_i)))$
    $\omega = \frac{\omega_\theta}{\omega_x}$
  end if
  while not converged **do**
    for discriminator iterations **do**
      Sample mini-batch $\mathbf{X}_d$, Θ_d from $\mathbf{X}_i$, Θ_i
      $\mathbf{Z} \sim p_t(\mathbf{Z}) = p(\mathbf{Z})(\omega(f_\phi(\mathbf{Z}, \mathbf{X}_d), \mathbf{X}_d))^{-1}$
      $\hat{\Theta}_d = f_\phi(\mathbf{Z}, \mathbf{X}_d)$
      $L = \sum_{\mathbf{X}_d} (\sum_{\Theta_d} \log \mathrm{D}_\psi(\Theta_d, \mathbf{X}_d) + \sum_{\hat{\Theta}_d} \log(1 - \mathrm{D}_\psi(\hat{\Theta}_d, \mathbf{X}_d)))$
      $\psi \leftarrow \psi + \lambda \nabla_\psi L$
    end for
    for generator iterations **do**
      Sample mini-batch $\mathbf{X}_g$, Θ_g from $\mathbf{X}_i$, Θ_i
      $\mathbf{Z} \sim p_t(\mathbf{Z}) = p(\mathbf{Z})(\omega(f_\phi(\mathbf{Z}, \mathbf{X}_g), \mathbf{X}_g))^{-1}$
      $\hat{\Theta}_g = f_\phi(\mathbf{Z}, \mathbf{X}_g)$
      $L = -\sum_{\mathbf{X}_g} \sum_{\hat{\Theta}_g} \log(1 - \mathrm{D}_\psi(\hat{\Theta}_g, \mathbf{X}_g))$
      $\phi \leftarrow \phi + \lambda \nabla_\phi L$
    end for
  end while
  $\pi(\theta) = f_\phi(\theta; x_\mathrm{o})$
end for

Algorithm 3 Sequential GATSBI with inverse importance weights

Input: $\pi(\theta)$, simulator $p(x|\theta)$, classifier ω = 1, observation x_{o}, f_ϕ, D_ψ, learning rate λ
Output: Trained GAN networks f_{ϕ^*} and D_{ψ^*}

for $i = 1 \cdots$ number of rounds **do** $\Theta_i = \{\theta_1, \theta_2, \ldots, \theta_n\} \overset{\text{i.i.d}}{\sim} \pi(\theta)$ $\mathbf{X}_i = \{x_1, x_2, \ldots, x_n\} \sim p(x|\theta)$
  if $i > 1$ **then**:
    $\omega_\theta \leftarrow \max_{\omega_\theta} \log \sigma(\omega_\theta(\Theta_0)) + \log(1 - \sigma(\omega_\theta(\Theta_i)))$
    $\omega_x \leftarrow \max_{\omega_x} \log \sigma(\omega_x(\mathbf{X}_0)) + \log(1 - \sigma(\omega_x(\mathbf{X}_i)))$
    $\omega = \frac{\omega_\theta}{\omega_x}$
  end if
  while not converged **do**
    for discriminator iterations **do**
      Sample mini-batch $\mathbf{X}_d$, Θ_d from $\mathbf{X}_i$, Θ_i
      $\mathbf{Z} \sim p(\mathbf{Z})$
      $\hat{\Theta}_d = f_\phi(\mathbf{Z}, \mathbf{X}_d)$
      $L = \sum_{\mathbf{X}_d}(\sum_{\Theta_d} \log \mathrm{D}_\psi(\Theta_d, \mathbf{X}_d) + \sum_{\hat{\Theta}_d}(\omega(\hat{\Theta}_d, \mathbf{X}_d))^{-1} \log(1 - \mathrm{D}_\psi(\hat{\Theta}_d, \mathbf{X}_d)))$
      $\psi \leftarrow \psi + \lambda \nabla_\psi L$
    end for
    for generator iterations **do**
      Sample mini-batch $\mathbf{X}_g$, Θ_g from $\mathbf{X}_i$, Θ_i
      $\mathbf{Z} \sim p(\mathbf{Z})$
      $\hat{\Theta}_g = f_\phi(\mathbf{Z}, \mathbf{X}_g)$
      $L = -\sum_{\mathbf{X}_g} \sum_{\hat{\Theta}_g}(\omega(\hat{\Theta}_g, \mathbf{X}_g))^{-1} \log(1 - \mathrm{D}_\psi(\hat{\Theta}_g, \mathbf{X}_g))$
      $\phi \leftarrow \phi + \lambda \nabla_\phi L$
    end for
  end while
  $\pi(\theta) = f_\phi(\theta; x_{\mathrm{o}})$
end for

C ADDITIONAL RESULTS

C.1 POSTERIORS FOR BENCHMARK PROBLEMS

We show posterior plots for the benchmark problems: SLCP (Fig. 7) and Two Moons (Fig. 8). In both figures, panels on the diagonal display the histograms for each parameter, while the off-diagonal panels show pairwise posterior marginals, i.e., 2D histograms for pairs of parameters, marginalised over the remaining parameter dimensions.

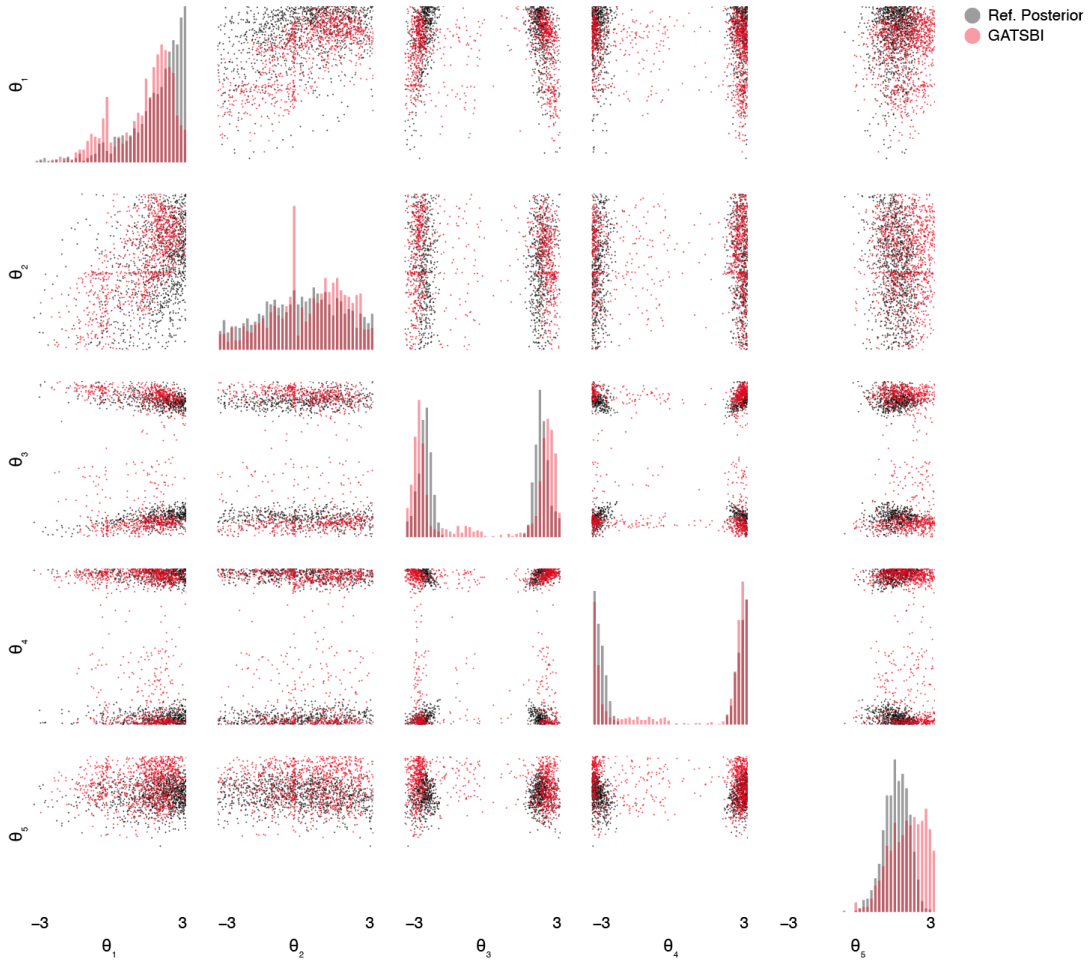

Figure 7: Inference for one test observation of the SLCP problem. Posterior samples for GATSBI trained on 100k simulations (red), and reference posterior samples (black). The GATSBI posterior samples cover well the disjoint modes of the posterior, although GATSBI sometimes produces samples in regions of low density in the reference posterior.

C.2 SEQUENTIAL GATSBI

We found that sequential GATSBI with the energy-based correction produced a modest improvement over amortised GATSBI for the Two Moons model with 1k and 10k simulation budgets, and no improvement at all with 100k (see Fig. 9). The inverse importance weights correction did not produce an improvement for any simulation budget. Sequential GATSBI performance was also sensitive to hyperparameter settings and network initialisation. We hypothesise that further improvement is possible with better hyperparameter or network architecture tuning.

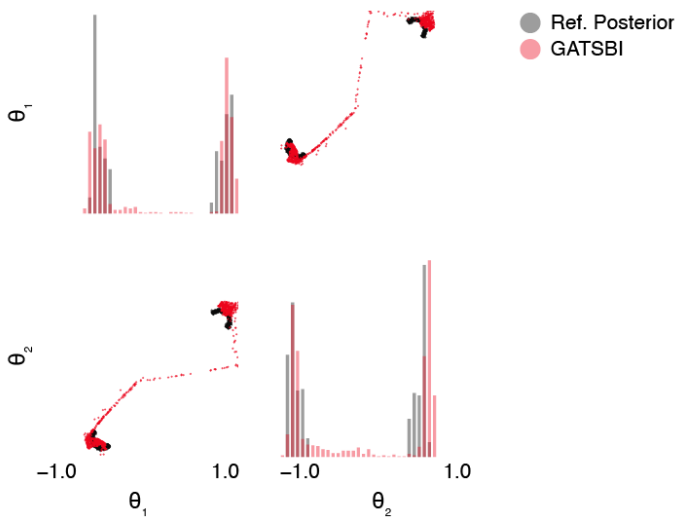

Figure 8: Inference for one test observation of the Two Moons problem. Posterior samples for GATSBI trained on 100k simulations (red), and reference posterior samples (black). GATSBI captures the global bi-modal structure in the reference posterior, but not the local crescent shape. It also generates some samples in regions of low density in the reference posterior.

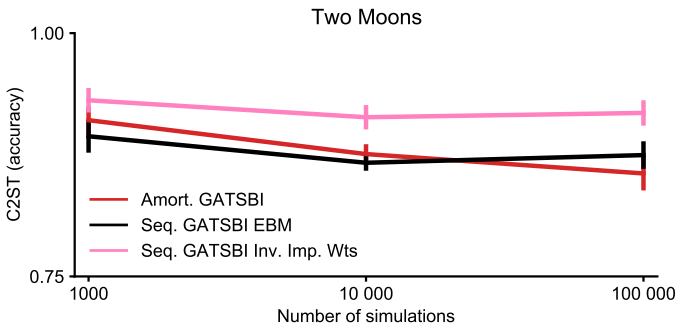

Figure 9: Sequential GATSBI performance for the Two Moons Model. The energy-based correction (EBM) results in a slight improvement over amortised GATSBI for 1k and 10k simulations, but the inverse importance weights correction does not.

D IMPLEMENTATION DETAILS

All networks and training algorithms were implemented in PyTorch (Paszke et al., 2019). We used Weights and Biases (Biewald, 2020) to log experiments with different hyperparameter sets and applications. We ran the high-dimensional experiments (camera model and shallow water model) on Tesla-V100 GPUs: the shallow water model required training to be parallelised across 2 GPUs at a time, and took about 4 days to converge and about 1.5 days for the camera model on one Tesla V100. We used RTX-2080Tis for the benchmark problems: the amortised GATSBI runs lasted a maximum of 1.5 days for the 100k budget; the sequential GATSBI runs took longer with the maximum being 8 days for the energy-based correction with a budget of 100k. On similar resources, NPE took about 1 day to train on the shallow water model, and 3 weeks and 2 days to train on the camera model. NPE took about 10 min for 100k simulations on both benchmark problems. SMC-ABC and rejection-ABC both took about 6s on the benchmark problems with a budget of 100k.

D.1 SIMPLE-LIKELIHOOD COMPLEX POSTERIOR (SLCP) AND TWO-MOONS

Prior and simulator For details of the prior and simulator, we refer to Lueckmann et al. (2021).

GAN architecture The generator was a 5-layer MLP with 128 hidden units in the first four layers. The final layer had output features equal to the parameter dimension of the problem. A leaky ReLU nonlinearity with slope 0.1 followed each layer. A noise vector sampled from a standard normal distribution (two-dimensional for Two Moons, and 5-dimensional for SLCP) was injected at the fourth layer of the generator. The generator received the observations as input to the first layer, used the first three layers to embed the observation, and multiplied the embedding with the injected noise at the fourth layer, which was then passed through the subsequent layers to produce an output of the same dimensions as the parameters. The discriminator was a 6-layer MLP with 2048 hidden units in the first five layers and the final layer returned a scalar output. A leaky ReLU nonlinearity with slope 0.1 followed each layer, except for the last, which was followed by a sigmoid nonlinearity. The discriminator received both the observations and the parameters sampled alternatively from the generator and the prior as input. Observations and parameters were concatenated and passed through the six layers of the network. For Deep Posterior Sampling, the discriminator did not have the final sigmoid layer.

Training details The generator and discriminator were trained in parallel for 1k, 10k and 100k simulations, with a batch size = $\min$(10% of the simulation budget, 1000). For each simulation budget, 100 samples were held out for validation. We used 10 discriminator updates for 1k and 10k simulation budgets, and 100 discriminator updates for the 100k simulation budget, per generator update. Note that the increase in discriminator updates for 100k simulations is intended to compensate for the reduced relative batch size i.e., 1000 batches = 0.01%. The networks were optimised with the cross-entropy loss. We used the Adam optimiser (Kingma and Ba, 2015) with learning rate=0.0001, β_1=0.9 and β_2=0.99 for both networks. We trained the networks for 10k, 20k and 20k epochs for the three simulation budgets respectively. To ensure stable training, we used spectral normalisation (Miyato et al., 2018) for the discriminator network weights. For the comparison with LFVI, we kept the architecture and hyperparameters the same as for GATSBI, but trained the generator to minimise $L(\phi) = \mathbb{E}_{q_\phi(\theta|x)}[\log \frac{1-\mathrm{D}_\psi(\theta,x)}{\mathrm{D}_\psi(\theta,x)}]$. Similarly, for Deep Posterior Sampling, we kept the same architecture (minus the final sigmoid layer for the discriminator) and hyperparameters, but trained the generator and discriminator on the Wasserstein loss: $L(\phi,\psi) = \mathbb{E}_{p(\theta|x)}\,\mathrm{D}_\psi(\theta) - \mathbb{E}_{q_\phi(\theta|x)}\,\mathrm{D}_\psi(\theta)$.

Optimised hyperparameters for Two Moons model The generator was a 2-layer MLP with 128 hidden units in the first layer and 2 output features in the second layer (same as the parameter dimension). Each layer was followed by a leaky ReLU nonlinearity with slope 0.1. Two-dimensional white noise was injected into the second layer, after it was multiplied with the output of the first layer. The discriminator was a 4-layer MLP with 2048 hidden units in the first 3 layers each followed by a leaky ReLU nonlinearity of slope 0.1, and a single output feature in the last layer followed by a sigmoid nonlinearity. We trained the networks in tandem for approximately 10k, 50k and 25k epochs for the 1k, 10k and 100k simulation budgets respectively. There were 10 discriminator updates and 10 generator updates per epoch, with the batch size set to 10% of the simulation budget (also for

100k simulations). All other hyperparameters (learning rate, β_1, β_2, etc.) were the same as for the non-optimised architecture.

Hyperparameters for sequential GATSBI The architecture of the generator and discriminator were the same as for amortised GATSBI. We trained the networks for 2 rounds. In the first round, the networks were trained with the same hyperparameters as for amortised GATSBI, on samples from the prior: the only exceptions were the number of samples we held out for the 1k budget (10 instead of 100) and the number of discriminator updates per epoch for the 100k budget (10 instead of 100). Both exceptions were to ensure that there were always 10 discriminator updates per epoch, and speeding up training as much as possible. In the second round, the networks were trained using the energy-based correction with samples from the proposal prior, as well as samples from the prior used in the first round. All other hyperparameters were the same as for round one. The simulation budget was split equally across the two rounds, and the networks were trained anew for each of 10 different observations. The number of epochs was the same for the first and second round: 5k, 10k and 20k for the 1k, 10, and 100k budget respectively. We trained 2 classifiers at the beginning of round two: one to approximate the ratio $\frac{\pi(\theta)}{\tilde{\pi}(\theta)}$ and the other to approximate $\frac{p(x)}{\tilde{p}(x)}$. Both classifiers were 4-layer MLPs with 128 hidden units in each layer, and a ReLU nonlinearity following each layer. The classifiers were trained on samples from the proposal prior and prior, and the proposal marginal likelihood and likelihood respectively, using the MLPClassifier class with default hyperparameters (except for "max_iter"=5000) from scikit-learn (Pedregosa et al., 2011). For the energy-based correction, we used rejection sampling to sample from the corrected distribution $p_t(z)$: for a particular observation x, we sampled z $p(z) = \mathcal{N}(0, 1)$, evaluated the probability of acceptance $p = (\omega(f_\phi(z, x), z))^{-1}/M$, and accepted z if $u < p$ where u is a uniform random variable. To compute the scale factor M, we simply took the maximum value of $p(z)\omega(f_\phi(z, x), z))^{-1}$ within each batch. For the inverse importance weights correction, we computed $(\omega(\theta, x))^{-1}$, used it to calculate the loss as in Equation equation 21 for each discriminator and generator update in the second round.

D.2 SHALLOW WATER MODEL

Prior $\theta \sim \mathcal{N}(\mu\mathbf{1}_{100}, \Sigma),\ \theta \in \mathbb{R}^{100}$
$\mu = 10, \Sigma_{ij} = \sigma \exp(-(i-j)^2/\tau), \sigma = 15, \tau = 100.$

The values for $\boldsymbol{\mu}$ and Σ were chosen to ensure that the different depth profile samples produced discernible differences in the corresponding simulated surface waves, particularly in Fourier space. For example, combinations of μ values > 25 (deeper basins), σ values < 10 and τ values > 100 (smoother depth profiles) resulted in visually indistinguishable surface wave simulations.

Simulator

$$\mathbf{x}|\theta = f(\theta) + 0.25\boldsymbol{\epsilon}$$
$$\boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_{1,1} & \dots & \epsilon_{1,100} \\ \vdots & \ddots & \vdots \\ \epsilon_{200,1} & \dots & \epsilon_{200,100} \end{bmatrix} \quad \epsilon_{ij} \sim \mathcal{N}(0, 1).$$

$f(\theta)$ is obtained by solving the 1D Saint-Venant equations on a 100-element grid, performing a 2D Fourier transform and stacking the the real and imaginary part to form a $2\times100\times100$-dimensional array.

The equations were solved using a semi-implicit solver (Backhaus, 1983) with a weight of 0.5 for each time level, implemented in Fortran (F90). The time-step size dt was set to 300s and the simulation was run for a total of 3600s. The grid spacing dx was 0.01, with dry cells at both boundaries using a depth of -10. We used a bottom drag coefficient of 0.001 and gravity=$9.81^m/_{s^2}$. An initial surface disturbance of amplitude 0.2 was injected at $x = 2$, to push the system out of equilibrium.

We chose to perform inference with observations in the Fourier domain for the following reason. Since waves are a naturally periodic delocalised phenomenon, it makes sense to run inference on their Fourier-transformed amplitudes, so that convolutional filters can pick up on localised features. We used the scipy fft2 package (Virtanen et al., 2020).

GAN architecture The generator network was similar to the DCGAN generator (Radford et al., 2015). There were five sequentially stacked blocks of the following form: a 2D convolutional layer, followed by a batch-norm layer and ReLU nonlinearity, except for the last layer, which had only a convolutional layer followed by a $\tanh$ nonlinearity. The observations input to the generator were of size batch size $\times$ 200 $\times$ 100. The input channels, output channels, kernel size, stride and padding for each of the six convolutional layers were as follows: 1 - (2, 512, 4, 1, 0), 2 - (512, 256, 4, 2, 1), 3 - (256, 128, 4, 2, 1), 4 - (128, 128, 4, 2, 1), 5 - (128, 1, 4, 2, 1). The final block was followed by a fully-connected readout layer that returned a 100-dimensional vector. Additionally, we sampled 25-dimensional noise, where each element of the array was drawn independently from a standard Gaussian. and added it to the output of the $\tanh$ layer, just before the readout layer.

The discriminator network was similar to the DGCAN discriminator: it contained embedding layers that mirrored the generator network minus the injected noise. The input channels, output channels, kernel size, stride and padding for each of the six convolutional layers in the embedding network were as follows: 1 - (2, 256, 4, 1, 0), 2 - (256, 128, 4, 2, 1), 3 - (128, 64, 4, 2, 1), 4 - (64, 64, 4, 2, 1), 5 - (64, 1, 4, 2, 1). This is followed by 4 fully-connected layers with 256 units each and a leaky ReLU nonlinearity of slope 0.2 after each fully-connected layer. The final fully-connected layer, however, was followed by a sigmoid nonlinearity. The discriminator received both the Fourier-transformed waves and a depth profile alternatively from the generator and the prior as input. The Fourier-transformed waves were passed through the embedding layers, the embedding was concatenated with the input depth profile and then passed through the fully-connected layers.

Training details The two networks were trained in parallel for $\sim$40k epochs, with 100k training samples, of which 100 were held out for testing. We used a batch size of 125, the cross-entropy loss and the Adam optimiser with learning rate$= 0.0001$, $\beta_1 = 0.9$ and $\beta_2 = 0.99$ for both networks. In each epoch, there was 1 discriminator update for every generator update. To ensure stability of training, we used spectral normalisation for the discriminator weights, and clipped the gradient norms for both networks to 0.01, unrolled the discriminator(Metz et al., 2017) with 10 updates i.e., in each epoch, we updated the discriminator 10 times, but reset it to the state after the first update following the generator update.

NPE and NLE We trained NPE and NLE as implemented in the sbi package (Tejero-Cantero et al., 2020) on the shallow water model. We set training hyperparameters as described in Lueckmann et al. (2021), except for the training batch size which we set to 100 for NPE and NLE. The number of hidden units in the density and ratio estimators which we set to 100 (default is 50). For NPE, we included an embedding net to embed the 20k-dimensional observations to the number of hidden units. This embedding net was identical to the one used for the GATSBI discriminators, and it was trained jointly with the corresponding density or ratio estimators. We trained with exactly the same 100k training samples used for GATSBI. MCMC sampling parameters for NLE were set as in Lueckmann et al. (2021).

To calculate **correlation coefficients** for the GATSBI and NPE posteriors, we sampled 1000 depth profiles from the trained networks for each of 1000 different observations from a test set. We then calculated the mean of the 1000 depth profile samples per observation, and computed the correlation coeffiecient of the mean with the corresponding groundtruth depth profile. Thus, we had 1000 different correlation coefficients; we report the mean and the standard deviation for these correlation coefficients.

Simulation-based calibration (Talts et al., 2020) offers a way to evaluate simulation-based inference in the absence of ground-truth posteriors. SBC checks whether the approximate posterior $q_\phi(\theta|x)$, when marginalised over multiple observations x, converges to the prior $\pi(\theta)$. A posterior that satisfies this condition is well-calibrated, although it is not a sufficient test of the quality of the learned posterior, since a posterior distribution that is equal to the prior would also be well-calibrated. However, when complemented with posterior predictive checks it provides a good test for intractable inference problems.

We performed SBC on the shallow water model. To obtain a test data set $\{\theta_i, x_i\}_{i=1}^N$, we sampled $N = 1000$ parameters θ_i from the prior and generated corresponding observations x_i from the simulator. For each x_i, we then obtained a set of $L = 1000$ posterior samples using the GATSBI generator and calculated the rank of the test parameter θ_i under the L GATSBI posterior samples

as described in algorithm 1 in Talts et al. (2020), separately for each posterior dimension. For the ranking we used a Gaussian random variable with zero mean and variance 10. We then used bins of $n = 20$ to compute and plot the histogram of the rank statistic. According to SBC, if the marginalised approximate posterior truly matched the prior, the rank statistic for each dimension should be uniformly distributed. Performing SBC can be computationally expensive because the inference has to be repeated for every test data point. In our scenario it was feasible only because GATSBI and NPE perform amortised inference and do not require retraining or MCMC sampling (as in the case of NLE) for every new x. We followed the same procedure to do SBC on NPE as for GATSBI.

D.3 NOISY CAMERA MODEL

Prior The parameters θ were 28×28-dimensional images sampled randomly from the 800k images in the EMNIST dataset.

Simulator The simulator takes a clean image as input, and corrupts it by first adding Poisson noise, followed by a convolution with a Gaussian point-spread function: $\mathbf{m} \sim \text{Poisson}(\theta)$
$\mathbf{x}|\mathbf{th} = f * m; \quad f(t) = \exp(-\frac{t^2}{\sigma^2}))$ where $*$ denotes a convolution operation with a series of 1D Gaussian filters given by f. We set the width of the Gaussian function $\sigma = 3$.

GAN architecture The generator network was similar to the Pix2Pix generator (Isola et al., 2016): there were 9 blocks stacked sequentially; the first 4 blocks consisted of a 2D convolutional layer, followed by a leaky ReLU nonlinearity with slope 0.2 and a batchnorm layer; the next 4 blocks consisted of transpose a convolutional layer, followed by a leaky ReLU layer of slope 0.2 and a batchnorm layer; the final block had a transposed convolutional layer followed by a sigmoid nonlinearity. The input channels, output channels, kernel size, stride and padding for each of the convolutional or transposed convolutional layers in the 9 blocks were as follows: 1 - (1, 8, 2, 2, 1), 2 - (8, 16, 2, 2, 1), 3 - (16, 32, 2, 2, 1), 4 - (32, 64, 3, 1, 0), 5 - (128, 32, 3, 2, 1), 6 - (64, 16, 2, 2, 1), 7 - (32, 8, 3, 2, 1), 8 - (16, 4, 2, 2, 1), 9 - (4, 1, 1, 1, 0). There were skip-connections from block 1 to block 8, block 2 to block 7, block 3, to block 6 and from block 4 to block 5. 200-dimensional white noise was injected into the fifth block, after convolving it with a 1D convolutional filter and multiplying it with the output of the fourth block.

The discriminator network was again similar to the Pix2Pix discriminator: we concatenated the image from the generator or prior with the noisy image from the simulator, and passed this through 4 blocks consisting of a 2D convolutional layer, a leaky ReLU nonlinearity of slope 0.2 and a batch-norm layer, and finally through a 2D convolutional layer and a sigmoid nonlinearity. The input channels, output channels, kernel size, stride and padding for each of the convolutional were as follows: 1 - (2, 8, 2, 2, 1), 2 - (8, 16, 2, 2, 1), 3 - (16, 32, 2, 2, 1), 4 - (32, 64, 2, 2, 1), 5 - (64, 1, 3, 1, 0).

Training details The generator and discriminator were trained in tandem for 10k epochs, with 800k training samples, of which 100 were held out for testing. We used a batch size of 800, the cross-entropy loss and the Adam optimiser with learning rate$= 0.0002$, $\beta_1 = 0.5$ and $\beta_2 = 0.99$ for both networks. In each epoch, there was a single discriminator update for every second generator update. To ensure that training was stable, we used spectral normalisation for the discriminator weights, and clipped the gradient norms for both networks to 0.01.

NPE We trained NPE using the implementation in the sbi package (Tejero-Cantero et al., 2020). We set training hyperparameters as described in Lueckmann et al. (2021), except for the training batch size which we set to 1 (in order to ensure that we did not run out of memory while training), and the number of hidden units in the density estimators which we set to 100. 28×28 dimensional observations were passed directly to the flow, without an embedding net or computing any summary statistics, as was also the case for GATSBI. We trained with exactly the same 800k training samples used for GATSBI.

REFERENCES

Jonas Adler and Ozan Öktem. Deep bayesian inversion, 2018.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 214–223. PMLR, 06–11 Aug 2017.

Samaneh Azadi, Catherine Olsson, Trevor Darrell, Ian Goodfellow, and Augustus Odena. Discriminator Rejection Sampling. *arXiv:1810.06758 [cs, stat]*, 2019. arXiv: 1810.06758.

Jan O Backhaus. A semi-implicit scheme for the shallow water equations for application to shelf sea modelling. *Continental Shelf Research*, 2(4):243–254, 1983.

Lukas Biewald. Experiment tracking with weights and biases, 2020. Software available from wandb.com.

Tong Che, Ruixiang Zhang, Jascha Sohl-Dickstein, Hugo Larochelle, Liam Paull, Yuan Cao, and Yoshua Bengio. Your GAN is Secretly an Energy-based Model and You Should use Discriminator Driven Latent Sampling. *arXiv:2003.06060 [cs, stat]*, 2020.

Jeff Donahue, Trevor Darrell, and Philipp Krähenbühl. Adversarial feature learning. *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, pages 1–18, 2019.

Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, and Aaron Courville. Adversarially learned inference. *arXiv preprint arXiv:1606.00704*, 2016.

William Fedus, Mihaela Rosca, Balaji Lakshminarayanan, Andrew M Dai, Shakir Mohamed, and Ian Goodfellow. Many paths to equilibrium: Gans do not need to decrease a divergence at every step. *arXiv preprint arXiv:1710.08446*, 2017.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

Ferenc Huszár. Variational inference using implicit distributions. *arXiv preprint arXiv:1702.08235*, 2017.

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *arXiv preprint arXiv:1611.07004*, 2016.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations, ICLR*, 2015.

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 30*, pages 1289–1299. Curran Associates, Inc., 2017.

Jan-Matthis Lueckmann, Jan Boelts, David Greenberg, Pedro Goncalves, and Jakob Macke. Benchmarking simulation-based inference. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, pages 343–351, 2021.

Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*, 2015.

Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. Adversarial variational Bayes: Unifying variational autoencoders and generative adversarial networks. *34th International Conference on Machine Learning, ICML 2017*, 5:3694–3707, 2017.

Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge?, 2018.

Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks, 2017.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *CoRR*, abs/1802.05957, 2018.

George Papamakarios, David Sterratt, and Iain Murray. Sequential neural likelihood: Fast likelihood-free inference with autoregressive flows. In *Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 89 of *Proceedings of Machine Learning Research*, pages 837–848. PMLR, 2019.

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 32*, pages 8024–8035. Curran Associates, Inc., 2019.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Guo-Jun Qi. Loss-sensitive generative adversarial networks on lipschitz densities, 2018.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Sean Talts, Michael Betancourt, Daniel Simpson, and Aki Vehtari. Validating Bayesian Inference Algorithms with Simulation-Based Calibration. pages 1–26, 2020.

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.

Dustin Tran, Rajesh Ranganath, and David Blei. Hierarchical implicit models and likelihood-free variational inference. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

The first rule of synaptic plasticity: there is no one rule that explains your data

Poornima Ramesh
University of Tübingen

Basile Confavreux
Institute of Science and Technology Austria

Pedro P. J. Gonçalves
University of Tübingen

Tim P. Vogels*
Institute of Science and Technology Austria

Jakob H. Macke*
University of Tübingen
Max Planck Institute for Intelligent Systems

* **Equal contribution**

Abstract

Synaptic plasticity is thought to be critical for building and maintaining the neuronal networks that support mammalian behavior and cognition. Models of plasticity are typically designed by hand. Starting with specific hypothesized rules to change synaptic efficacy, models are then evaluated based on the similarity of their effects to empirical data. While this approach has provided some good insight into plasticity mechanisms in the past, it is limited by human intuition. It cannot infer plasticity rules from data directly, nor identify whether alternative rules may explain observations similarly well. Here, we provide a new approach that uses Generative Adversarial Networks for learning plasticity rules directly from data. This approach allows us to effectively explore the space of plausible plasticity mechanisms consistent with the observations. We validate our method by re-discovering Oja's rule from simulated postsynaptic activity traces. We then show that expanding the search space for plasticity rules with different parametrizations leads to qualitatively different learned rules. Despite their differences, each unique rule generates neural activity which is statistically similar to that of the ground-truth rule. Our results signal the need for a shift in how we study plasticity rules—instead of focusing on a small number of individual rules, we should identify *families* of rules which lead to similar network function, and characterize their shared computational and mechanistic features for a full understanding of how autonomous rules build and maintain the substrate of our thinking.

1 Introduction

Synaptic plasticity is the key to the brain's ability to learn from and remember past experiences [Abbott and Nelson, 2000, Citri and Malenka, 2008]. However, the exact nature of local synaptic changes and how they relate to network-level properties remain largely unknown [Magee and Grienberger, 2020, Morrison et al., 2007]. This is due in part to the difficulty in observing directly *in vivo* how populations of synapses evolve across time. As a result, the default avenue to link putative plasticity mechanisms to available data and hypotheses has long been theoretical work Gerstner et al. [1996], Pfister and Gerstner [2006], Clopath et al. [2010], Graupner and Brunel [2012].

Preprint. Under review at 36th Conference on Neural Information Processing Systems 2022.

Theoretical models of synaptic plasticity typically encapsulate the various molecular contributions at individual synapses into a scalar "weight" associated with each synapse. The equations governing the evolution of said synaptic weights across time are referred to as plasticity rules. Classically, plasticity rules have been derived from analytical arguments or hypotheses inspired by experimental observations [Gerstner et al., 1996, Pfister and Gerstner, 2006, Clopath et al., 2010, Vogels et al., 2011]. Critically, since these approaches rely on human intuition and hand-tuning to conceive a *single* plasticity rule with a desired network effect (e.g. [Zenke et al., 2015, Litwin-Kumar and Doiron, 2014]), these studies cannot explore whether alternative plasticity mechanisms could also explain the data. Using such "unicorn rules", i.e., rules that produce observed results without regard for the biological fidelity of their ingredients, may lead to potentially erroneous experimental predictions. In other words, classical theoretical studies can only investigate whether a specific rule is *sufficient* but not whether it is *necessary* to account for observed phenomena, or whether it can stand up against the onslaught of reality in biological settings. A more comprehensive approach may be to algorithmically explore (or "meta-learn") the space of plausible plasticity rules subserving given observations or computations, assuming that the experimental data are rich enough to feed such explorations, and that sufficiently powerful numerical approaches are available.

Experimental data that are rich enough for meta-learning approaches have become more accessible in recent years, along with a boost in available computational power and improvements in numerical analysis. Progress in large-scale recording technologies [Jun et al., 2017, Nguyen et al., 2016] can deliver recorded neural activity from large populations of neurons, potentially providing the kind of datasets to derive plasticity rules from. However, success relies on: (i) a sufficiently flexible parametrization of the plasticity rule, so as to be able to fit the widest possible range of functions; and (ii) a measure of disparity between empirical and simulated data (i.e., a loss function), that is able to flexibly extract the data features that are informative about the underlying plasticity rule. Previous work focuses mainly on the former (i): Confavreux et al. [2020], Tyulmankov et al. [2022], Lindsey and Litwin-Kumar [2020], Jordan et al. [2021], Metz et al. [2018] design flexible parametrizations for their plasticity rules, but handcraft the loss function from network-level hypotheses so to deduce the exact form of the plasticity rule. Lim et al. [2015] specify a loss function on hand-picked data features. We propose to fulfill both conditions (i) and (ii) by deducing both the plasticity rule *and* loss function directly from data. To this end, we make full use of the flexibility of Deep Neural Networks (DNNs) to *both* parametrize plasticity rules, and to compare simulated and empirical data. More specifically, we formulate this question as an adversarial game in which DNNs aim to produce plasticity rules which in turn create simulated data that "tricks" the discriminator into misclassifiying it as empirical. Such use of Generative Adversarial Networks [GANs, Goodfellow et al., 2014] to meta-learn plasticity rules from observed data thus alleviates the need to handcraft either the plasticity rule or the loss function.

Our GAN framework infers plasticity rules from recorded activity traces. As a proof of concept, we start by applying our GAN approach to synthetic data simulated with a known plasticity rule, i.e., Oja's rule—a canonical plasticity rule [Oja, 1982]. This allows to compare the rules learned with our framework to the ground-truth plasticity rule. We show that the plasticity rules learned using our adversarial approach generate activity traces that are statistically similar to the data, even though the rules themselves are very different from Oja's rule, i.e., their synaptic weight dynamics differ substantially. Furthermore, we show that various model and data biases dramatically change the form of learned rules without affecting the quality of the fit to data. Our findings point towards a need to shift from studying plasticity with the aim of finding the unique "correct" rule responsible for a given network function, to attempting to infer entire families of rules with similar network-level function. Such sets of rules may comprise individual rules with different mechanistic implementations. Characterizing the conserved properties within such classes of equivalent rules will be key to producing more robust experimental predictions.

2 Methods

Below, we introduce the rate network model, Oja's rule: the rule underlying our ground-truth (synthetic) data, the different parametrizations for the learning rules and the GAN-based meta-learning framework.

Network model: We consider a linear feedforward rate network with N presynaptic neurons with activity $x_j, j = 1 \dots N$, M postsynaptic neurons with activity $y_i, i = 1 \dots M$, and synaptic weights ω_{ij}. The postsynaptic activity at (discretized) time t is updated as follows:

$$y_i^t = \sum_j \omega_{ij}^t x_j^t \tag{1}$$

Ground-truth data and Oja's rule: In lieu of experimental data, and to test our framework, ground-truth data consisted of activity traces from the rate network defined above, evolving with Oja's rule. This rule consists of a Hebbian term, with an added normalization term for stability:

$$\Delta\omega_{ij}^t = x_j^t * y_i^t - (y_i^t)^2 \omega_{ij}^t \tag{2}$$

This Hebbian plasticity rule, when used to update the synaptic weights ω_{ij} of the feedforward rate network in Eqn 1, causes ω_{ij} to converge to the first principal component PC_1 of the presynaptic activity $x = \{x_i\}_{i=1}^N$ [Oja, 1982]. Concurrently, under Oja's rule, the postsynaptic activity y converges to a projection of x onto its first principal component. In order to simulate activity traces y_i, we sampled presynaptic activity x from a N-dimensional Gaussian distribution, with a given covariance structure (details in Appendix Sec. B). Note that we fixed the presynaptic activity to be constant across time, and that the synaptic weight update $\Delta\omega_{ij}$ at each time step was computed by averaging over an ensemble of pre- and postsynaptic activities corresponding to $K = 100$ different x samples from the multivariate Gaussian distribution. In other words, we have implicit batch learning for the synaptic weights[1]:

$$\Delta\omega_{ij} = \frac{1}{K} \sum_{k=1}^{K} x_j^{(k)} * y_i^{(k)} - (y_i^{(k)})^2 w_{ij} \tag{3}$$

We simulated the plastic network for $T = 200$ time steps. The ground-truth data consisted of the postsynaptic activity at every time step for each of $K = 100$ different 3-dimensional presynaptic activities and randomly initialised synaptic weights $\omega_{ij}^0 \sim \mathcal{N}(0, 0.1^2)$.

In Fig 4, we used two other types of data: (1) we introduced noise in the postsynaptic activity: $y_i^t = \sum_j \omega_{ij}^t x_j^t + \epsilon, \epsilon \sim \mathcal{N}(0, 0.25^2)$; (2) 39 input neurons instead of 3 were simulated, the training was otherwise similar.

Parametrized learning rules In order to meta-learn the synaptic plasticity rules from the ground-truth data, we formalized the learning rule with a parametrized function h_θ:

$$\Delta\omega_{ij} = h_\theta(\omega_{ij}^t, x_j^t, y_i^t). \tag{4}$$

First, for a proof of principle (Fig. 2), we parametrized Oja's rule with learnable coefficients θ_1 and θ_2:

$$h_\theta(\omega_{ij}^t, x_j^t, y_i^t) = \theta_1 x_j^t * y_i^t + \theta_2 (y_i^t)^2 \omega_{ij}^t \tag{5}$$

For all other experiments, the plasticity rules were parametrized using multi-layer perceptrons [MLPs, Bishop et al., 1995], with different inputs from the rate network, depending on the chosen model:

- *Local MLP*: the plasticity rule is parametrized by a 3-layer MLP, i.e., $h_\theta(\cdot) = \mathrm{MLP}_\theta(\omega_{ij}^t, x_j^t, y_i^t)$. This MLP represents a *local update*, i.e., it transforms each x_j^t, y_i^t and ω_{ij}^t in the same way, independently of the indices i, j and t.
- *Oja + local MLP*: $h_\theta(\cdot) = \mathrm{MLP}_\theta(\omega_{ij}^t, x_j^t, y_i^t) + x_j^t * y_i^t - (y_i^t)^2 \omega_{ij}^t$. This learning rule is "biased" since, by construction, it is initialised close to the ground-truth solution and any non-zero outputs of the MLP are perturbations to Oja's rule.
- *Semi-global MLP* computes the synaptic weight update $\Delta\omega_{ij}^t$ for a single synapse. It takes into account the mean presynaptic activity and the mean across the network synaptic weights at the current time step, in addition to the local presynaptic activity x_j^t, synaptic weight ω_{ij}^t and postsynaptic activity y_i^t: $\Delta\omega_{ij}^t = h_\theta(\cdot) = \mathrm{MLP}_\theta(\omega_{ij}^t, x_j^t, y_i^t, \frac{1}{N}\sum_j x_j^t, \frac{1}{N}\sum_j \omega_{ij})$.
- *Global MLP* takes into account all pre- and postsynaptic activities and synaptic weights: $\Delta\omega_{ij} = h_\theta(\cdot) = \mathrm{MLP}_\theta(\{\omega_{ij}^t, x_j^t, y_i^t\}_{i=1,j=1}^{M,N})$.

Note that these parametrized rules are progressively less constrained, and that each MLP could, in principle, be reduced to the one above it.

[1] To lighten the notation, we elide the dependence on time t in this equation.

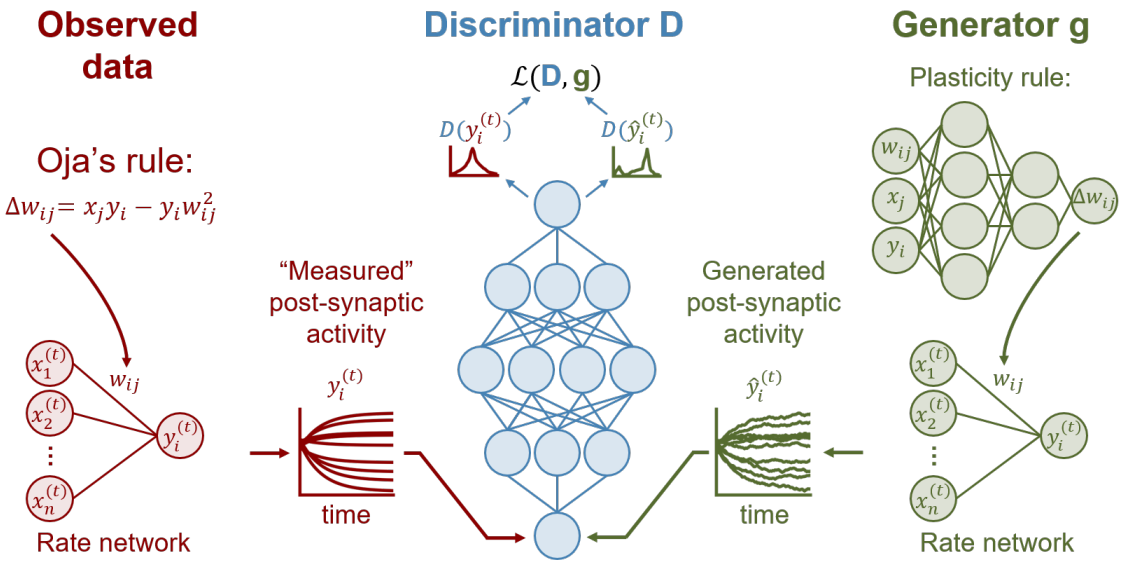

Figure 1: **Adversarial learning of plasticity rules.** Observed data are simulations of the postsynaptic activity of a rate network with plastic synapses evolving according to Oja's rule. The generator is a rate network with synapses evolving according to a learnable MLP rule. The discriminator is an expressive network that is trained to distinguish observed data from the output of the generator. In our framework, the generator and discriminator are trained so that at convergence, the learned MLP rule is such that the generator produces postsynaptic neural activities similar to the observed data.

Meta-learning framework Our meta-learning framework uses a GAN to learn the parameters θ of the plasticity rule h_θ, given neural activity. Below, we first introduce the GAN formalism, and then show how one can recast the problem of meta-learning within this formalism.

GANs are a machine learning approach to obtain generative models of data, by training a model to match a target distribution $p(d')$, which we only have access to via samples d' from the distribution. GANs consist of two deep neural networks: a generator network g_θ that produces data $d = g_\theta(z)$, by deterministically transforming latent random variables z sampled from a known distribution $p(z)$. The second network is a discriminator D_ψ which aims to classify generated samples d as fake (i.e., from the generator), and d' as real (i.e., from the target distribution, Fig. 1). After the two networks have been trained with a minimax loss

$$\theta^*, \psi^* = \min_\theta \max_\psi \mathbb{E}_{p(d')} \log \mathrm{D}_\psi(d') + \mathbb{E}_{p(z)} \log\big(1 - \mathrm{D}_\psi(g_\theta(z))\big), \tag{6}$$

in the limit of infinite data, the generator implicitly represents $p(d')$ at convergence. Thus, convergence yields a generative model of d'. Note that the GAN does not impose any restrictions on the architecture of the generator network. Furthermore, it does not define an explicit loss function for the target data d' and generated data d: instead, the discriminator implicitly represents a distance function between the two data distributions, and this function is also learned end-to-end with the generator. This leads to GANs not attempting to reproduce the data d' in minute detail, but rather to capture the general statistics of the data. In other words, the GAN matches the distributions $p(d)$ and $p(d')$, rather than individual samples d and d'.

This flexibility in the GAN framework is advantageous for meta-learning plasticity rules from neural activity. We first recast the system composed of the postsynaptic update (Eqn 1) and the plasticity rule (Eqn 4) as a generative model of postsynaptic activity traces for T time steps, i.e., $\mathbf{y} = y_{i\,i=1,t=1}^{t\,M,T} = g_\theta(\mathbf{x}, \omega^0)$, where $\mathbf{x} = \{x_j^t\}_{j=1,t=1}^{N,T}$ and $\omega^t = \{\omega_{ij}^t\}_{i=1,j=1}^{M,N}$ (Fig. 1). We then define a discriminator D_ψ that differentiates between generated activity traces $\mathbf{y}$ and observed activity traces $\mathbf{y}' \sim p(\mathbf{y}')$. Note that $\mathbf{y}$ is a random variable, since the rate network is initialised randomly before every forward pass from the generative model. We learn the parameters of the plasticity rule θ and the discriminator ψ using the same minimax loss as in Eqn 6:

$$\theta^*, \psi^* = \min_\theta \max_\psi \mathbb{E}_{p(\mathbf{y}')} \log \mathrm{D}_\psi(\mathbf{y}') + \mathbb{E}_{p(\omega^0)p(\mathbf{x})} \log\big(1 - \mathrm{D}_\psi(g_\theta(\mathbf{x}, \omega^0))\big). \tag{7}$$

At convergence, the GAN will have learned a plasticity rule h_θ that is consistent with the observed neural activity $\mathbf{y}'$. The GAN framework thus allows us to flexibly parametrize the plasticity rule. It also allows us to learn the rule from neural activity, without having to specify a loss function on the neural activity. Details on method implementation and numerical experiments are in Appendix Sec. B.

3 Results

We use an adversarial approach to flexibly deduce plasticity rules from neural activity, which requires neither pre-specified data features nor a loss function (Fig. 1). We perform all experiments on simulated data with a known underlying rule i.e., Oja's rule (Sec. 2). This enables us to compare GAN-based solutions to ground-truth, both at the level of activity traces and plasticity rules.

In the following we will first validate our method with a constrained parametrization of the synaptic plasticity rule, and then turn to a more flexible search space for plasticity rules. We show that our approach unveils a wide "family" of rules that are all consistent with the observed data. Interestingly, the learned rules vary substantially depending on how the plasticity rule is parametrized, as well as on the quality and quantity of data used for inferring these rules.

Proof of principle We first show that our adversarial approach is generally capable of meta-learning rules from neural activity and in principle, to 'rediscover' the exact rule that was used to generate the data. We used postsynaptic activity simulated with Oja's rule for training. We ran the GAN framework in a constrained setting where the only tunable parameters were the coefficients of the two monomials of Oja's rule (Eqn. 5).

The GAN approach approximately learned the original coefficients ($\theta_1 \approx 1$ and $\theta_2 \approx -1$). Thus, the GAN learned a constrained rule identical to Oja's rule (Fig. 2A-D), showing that the GAN setup is capable of learning the true underlying rule in principle.

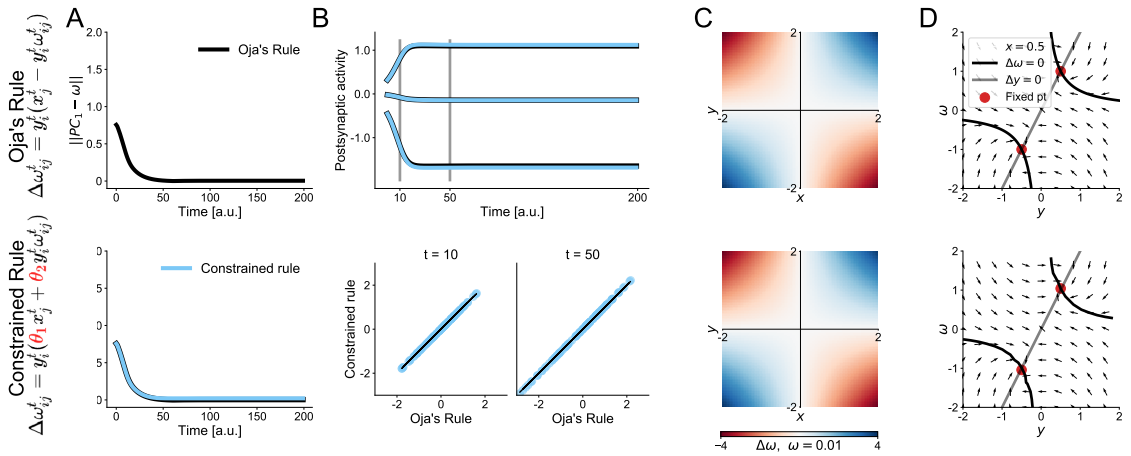

Figure 2: **GAN with constrained rule rediscovers Oja's rule.** Oja's rule (top, black), constrained rule (bottom, light blue). (A) Distance between the synaptic weights ω and the first principal component of $\mathbf{x}$ across time. (B) Postsynaptic activity traces for different initial synaptic weights in the rate network (top). Learned-rule activities versus the original Oja's rule activities at different time points and for different initial synaptic weights (bottom). (C) Synaptic weight updates $\Delta\omega$ for a range of presynaptic activities x and postsynaptic activities y and $\omega = 0.01$. (D) Vector field of ω versus postsynaptic activity y with presynaptic activity fixed at $x = 0.5$.

Rules learned from postsynaptic activity traces do not converge to Oja's rule We then considered a more flexible search space for the plasticity updates. We used the same observed data as before, but parametrized the plasticity rule using a 3-layer multi-layer perceptron (MLP) which updates each synapse in the linear network based on the presynaptic activity x_j^t, postsynaptic activity y_i^t and current synaptic weight of the given synapse ω_{ij}^t i.e., a local MLP rule (details in Sec. 2).

We found that the GAN learned a plasticity rule which accurately reproduces the statistics of the observed postsynaptic activity (Fig. 3C), after verifying that an MLP rule with random parameters, without any training could not reproduce observed postsynaptic activity (Appendix Fig. A.1). However, the dynamics of the synaptic weights from the GAN-learned rule are different from those of Oja's rule. We first quantified this with the Euclidean distance between the synaptic weights and the first principal component of the presynaptic activity PC_1 at each time point [Confavreux et al., 2020]. As expected, this distance decays to 0 with time for Oja's rule [Oja, 1982, Fig. 2A], but not for the learned MLP (Fig 3B), indicating that although the network activity is similar, the synaptic weights evolving with the MLP-rule do not converge to PC_1. This suggests that the synaptic weight dynamics are under-constrained by neural activity alone.

To visualize how the ground-truth and GAN-learned rules differed, we computed the synaptic weight update from each of the learned rules for a range of different presynaptic activities x_j and postsynaptic activities y_i—the observable network variables—, while the synaptic weights were fixed at $\omega_{ij} = 0.01$. The resulting synaptic weight-update diagrams $\Delta\omega_{ij} = f(x_j, y_i, \omega_{ij} = 0.01)$ appear qualitatively different for Oja's rule (Fig. 2C) and the learned MLP rule (Fig. 3D), both in terms of the magnitude of weight updates and regions of potentiation ($\Delta\omega_{ij} > 0$) and depression ($\Delta\omega_{ij} < 0$). For instance, Oja's rule is symmetric along the diagonal and anti-diagonal, whereas in the GAN-learned rule there is no symmetry, and $\Delta\omega_{ij} > 0$ only for a narrow range of $x_j - y_i$ values.

In order to have a more complete understanding of the dynamics of the GAN-learned rules, we computed the vector field of the postsynaptic activity and the synaptic weight, given a fixed presynaptic activity (Fig. 2D and Fig. 3E). While both rules have hyperbolic nullclines and two fixed points, in Oja's rule, both of the fixed points are stable, whereas the learned MLP has a stable node and a saddle node. The location of these fixed points is at higher values of ω for the learned MLP compared to Oja's rule. Overall, this indicates that the dynamics of the learned MLP rule and Oja's rule are distinct.

Thus, even though the learned rule reproduced the statistics of the ground-truth postsynaptic activity traces, the rule itself was qualitatively different from Oja's rule. We show that the conclusions illustrated here are representative of the behaviour for the entire family of datasets considered in this study (Appendix Fig. A.2): the mean squared error between the postsynaptic traces from Oja's rule and GAN-learned rules is relatively low for most test datasets.

Different inductive model biases lead to different GAN-learned rules We next explored how differences in model parametrizations (inductive biases) affected the learned rules. We hypothesized that models that depend on different variables would converge to different solutions.

To this end, in addition to the 3-layer MLP from above, we trained three plasticity rules: *Oja + local MLP* where the MLP learned perturbations to Oja's rule, *Semi-global MLP* which received the mean presynaptic activity and mean synaptic weight per timepoint in addition to the inputs for a local MLP, and *Global MLP* which received all presynaptic activity values, synaptic weights and postsynaptic activity per timepoint as input (Sec. 2 and Fig 3A) All three parametrized rules once trained, reproduced the statistics of the observed data (Fig 3C). In particular, the activity traces elicited by the Oja + local MLP rule are nearly identical to the traces from Oja's rule. The semi-global- and global MLP-generated activities capture relatively well the statistics of the observed postsynaptic activities. However the generated activity from the global MLP is slightly biased at later timepoints (Fig 3C). Nevertheless, it captures the salient features of the observed postsynaptic activity, e.g., the transient increase in postsynaptic activity at earlier timepoints, and the convergence to a stable value at later timepoints (Appendix Fig. A.3).

However, the weight trajectories for all three rules are different from Oja's Rule and from the local MLP (Fig. 3B). Upon further inspection of the weight-update diagrams and vector fields (Fig. 3D,E), Oja + local MLP has nearly the same dynamics as Oja's rule, which is consistent with the heavy bias introduced towards Oja's rule. Further analysis of the Oja + local MLP rule suggests that the MLP component contributes minimally to both the weight updates and the generated postsynaptic activity (Appendix Fig. A.4). Furthermore, the differences in the weight dynamics between Oja's rule and Oja + local MLP are due to the behaviour of the MLP in regions of the phase-space far away from the fixed points of Oja's rule (Appendix Fig. A.5). In contrast, both the semi-global MLP and global MLP have weight update diagrams and dynamics qualitatively different from each other and from Oja's rule (Fig. 3D,E). In particular, the GAN-learned rules have only one fixed point: stable in the

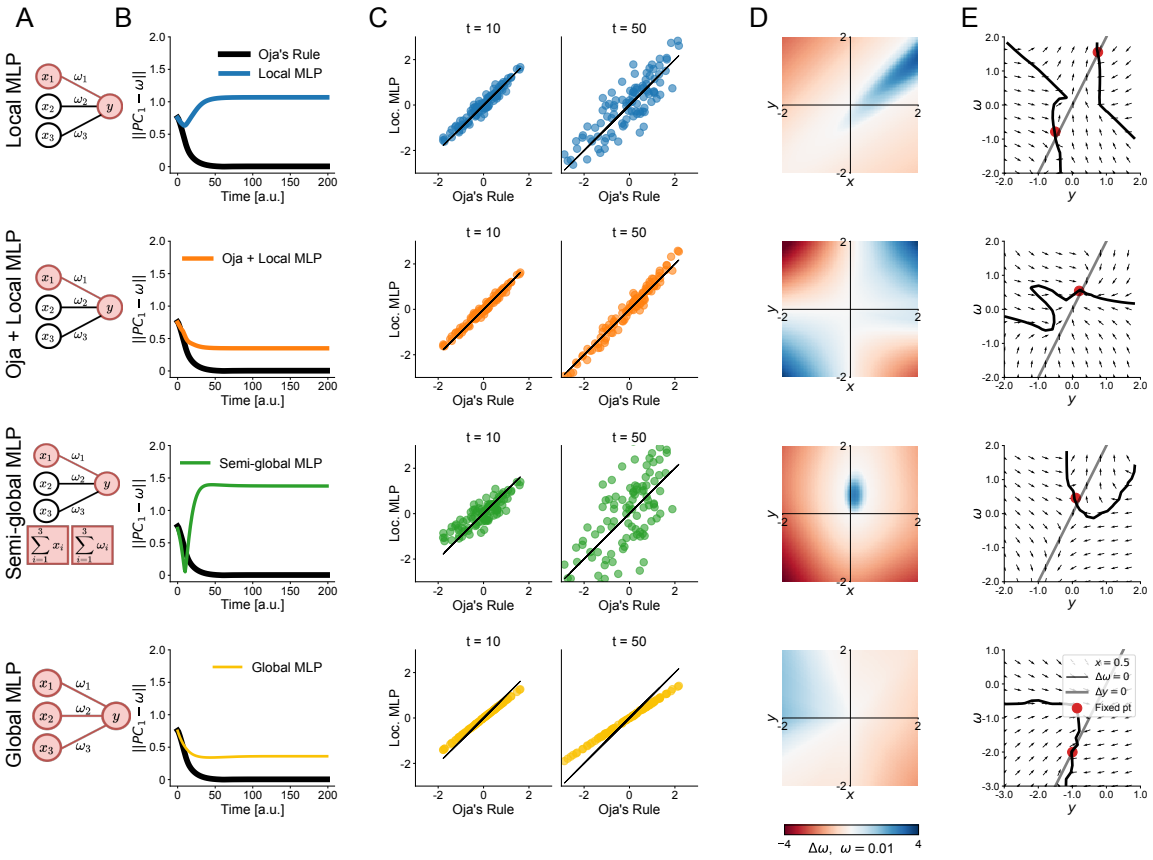

Figure 3: **Different model biases can lead to differences in the GAN-learned rules.** (A) Parametrized plasticity rules. Local MLP (top), Oja+Local MLP (second from top), semi-global MLP (second from bottom) and global MLP (bottom). (B) Weight trajectories, as measured by $||PC_1 - \omega||$ for Local MLP (top, blue), Oja + local MLP (second from top, orange), semi-global MLP (second from bottom, green), global MLP (bottom, yellow) and Oja's rule (black). (C) Observed and generated activity per time step. (D) Weight updates $\Delta\omega$ for learned rules. (E) Vector fields of ω versus postsynaptic activity y for learned rules.

case of the global MLP, and a saddle node in the case of semi-global MLP (Fig. 3E). Taken together, these results suggest that:

- There exists a whole family of mechanistically different plasticity rules that explain the postsynaptic activity equally well.
- Constraining or broadening our search space for these rules by specifying different parametrization leads to different learned rules.

In other words, there exist multiple local minima in the landscape of learning rules that are consistent with a particular set of observed data. We can think of different model architectures as placing the parametrized rule close to different minima in the rule landscape, resulting in convergence to different plasticity rules depending on the model architecture.

Different data biases lead to differences in the GAN-learned rules Finally, we explored how differences in training data influenced the learned rules. We perturbed the features of the postsynaptic activity (Fig 4A) by either (1) adding noise to it or (2) increasing the number of presynaptic neurons in the rate network (in our case, from 3 to 39, see Sec. 2 for details). We hypothesized that this would change the nature of the information available to the GAN discriminator to differentiate between observed and generated traces. This, in turn, would lead to different updates to the parametrized rules during training, and therefore, potentially different learned rules at convergence.

To this end, we trained the 3-layered local MLP architecture with data from the two other datasets. Note that the underlying plasticity rule for generating the datasets (1) and (2) was still Oja's rule. We tested the GAN-learned plasticity rules trained on dataset (1) and (2) on the same 3-neuron noiseless dataset used in the previous experiments (additional analysis on datasets (1) and (2) in Appendix Fig. A.6). Note that this was possible because the learned rules were all *local* and therefore, agnostic to the number of neurons in the rate network, and to the features of the postsynaptic activity of the observed data. We found that the two GAN-learned rules accurately reproduce the statistics of

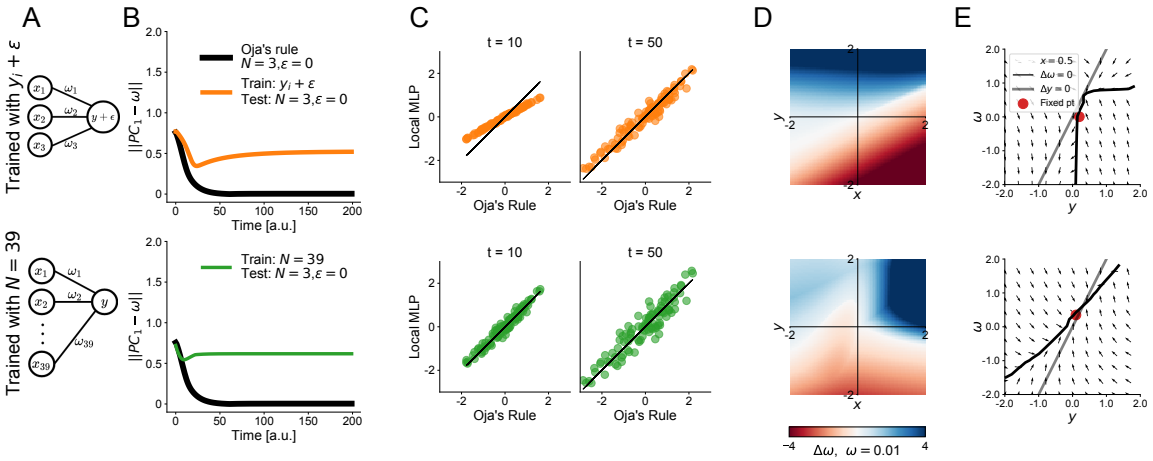

Figure 4: **Different data biases lead to differences in the GAN-learned rules.** Local rule trained on noisy postsynaptic neuron with 3 presynaptic neurons (top). Local rule trained on noiseless postsynaptic neuron with 39 presynaptic neurons (bottom). (A) Rate network architecture for different training data. (B) Weight trajectories for a network with 3 presynaptic neurons and a noiseless postsynaptic neuron given the two learned plasticity rules (orange and green). Oja's rule in black. (C) Learned MLP rules capture the statistics of activity from a rate network with Oja's rule. (D) Weight updates $\Delta\omega$ for the learned rules. (E) Vector fields of ω versus postsynaptic activity y for learned rules. Top: saddle node (red dot) and hyperbolic ω-nullcline (black line). Bottom: stable node and linear ω-nullcline.

the ground-truth activity traces from the out-of-training distribution (Fig. 4C), with the generated traces strongly correlated with the ground-truth traces for the same initial synaptic weights. We note that for the rule trained with noisy postsynaptic activity, the correlation is slightly biased at $t = 10$, but improves for $t = 50$. However, the weight update diagrams and the weight dynamics of the two learned rules once again differ substantially both from Oja's rule and the local MLP (4B,D,E). Overall, these findings imply that:

- Data biases can strongly influence the GAN-learned rules, potentially nudging the rules towards different minima in the landscape of learning rules, and resulting in different mechanistic predictions.
- The learned rules are scale-independent. Indeed, testing learned rules on a dataset with a different size that they have been trained on leads to similar results.
- Learned rules generalize to out-of-training data and retain meaningful features from the training data, i.e., the variety of rules observed are not the result of overfitting on the training data.

In summary, we showed that we can use a GAN framework to find a multitude of rules that are consistent with a activity traces simulated with Oja's rule. Vastly different rules, that did not resemble one another nor Oja's rule, were obtained under different perturbations to both data and plasticity search space. This suggests that the formulation of the problem and the various biases introduced by the chosen model and training data had arguably more impact on the putative experimental predictions made with the GAN-learned rules than the initial data itself.

4 Discussion

Theoretical studies about synaptic plasticity typically postulate specific network functions and hand-design rules that produce biological network behaviours similar to those observed in biological systems. Most research focuses on a small number of hand-designed canonical rules, with few terms capturing simple dependencies on network variables [Gerstner and Kistler, 2002, Pfister and Gerstner, 2006, Vogels et al., 2011]. However, each specific rule provides one of potentially many possible explanations of synaptic plasticity that are consistent with an observed dataset. In the absence of algorithmic approaches for *discovering* plasticity rules from data, it has been impossible to characterize how big the space of alternative solutions is. In principle, there could exist a multitude, or even whole families of such solutions, all of which explain observed data equally well. Without exploring this space of solutions, it is difficult to reason about the plasticity mechanisms that underlie memory consolidation and learning in the brain, and to have confidence in the robustness of the predictions derived from a single rule.

Here, we introduced a GAN framework that allows us to flexibly explore this space of solutions by algorithmically identifying plasticity rules from data. Using this approach, we showed how even in a very simple model, i.e., a rate network with Oja's Rule, a multitude of rules capture the same neural activities, and that these rules are sensitive to perturbations in data and model architecture. We emphasize that this multiplicity of rules occurred even under strongly idealized scenarios—in particular, we assumed the system to be stationary (other than the effect of plasticity), and that it could be observed without observation noise or sub-sampling of the neural population.

GANs have appealing attributes for our problem beyond the flexibility to approximate complex functions. In particular, and in our concrete application, GANs attempt to capture *distributions* of activity traces rather specific traces. This could potentially ensure that the learned rules do not overfit to trial-specific features in experimental data which are independent of synaptic plasticity, such as those related to the attentional state of an animal or changes in network input. The discriminator would be crucial in ignoring such idiosyncrasies with respect to the underlying plasticity rule. Our experiments are a preliminary indication that GANs are effective in this regard. Our learned rules show out-of-training generalization, with robustness both to changes in the rate network scale and perturbations to the rate network activity (Fig. 4).

Our framework could be extended beyond the current setting and more fully explore the flexibility of GANs as function approximators. In particular, GANs can accommodate conditioning on particular variables of interest [Mirza and Osindero, 2014]. For instance, this would allow the GAN to learn synaptic plasticity rules that are consistent with neural population activity conditioned on experimental settings or other variables external to the system of study, and thus potentially allowing for the GAN-learned rules to be context-dependent. Moreover, our approach to meta-learning rules with GANs shares similarities with unsupervised reinforcement learning [Gupta et al., 2018] where policy functions are meta-learned, neural ordinary differential equations [Chen et al., 2018] where the flow equations for ordinary differential equations are replaced with DNNs, and physics-informed GANs [Yang et al., 2018] where stochastic differential equations are learned using a GAN-framework. These similarities could be leveraged to improve our framework for meta-learning plasticity rules.

Nevertheless, there are limitations to our study. First, despite the flexibility to learn synaptic plasticity rules that GANs enable, their application can be technically demanding: they are notoriously hard to train, sensitive to initial conditions, and prone to mode collapse [Salimans et al., 2016, Srivastava et al., 2017, Huszár, 2015]. Second, our investigations focused on Oja's rule: while this is a canonical rule which has been extensively studied, it is likely too simple to form a realistic representation of plasticity mechanisms in the brain. In principle, it is conceivable that more complex rules (such as [Zenke et al., 2015, Litwin-Kumar and Doiron, 2014]) would be easier to infer from data unambiguously. Investigating this would require reproducing these results in more complex biological networks, ideally with experimental data. Furthermore, perturbative studies might help disambiguate multiple rules for the same data (although our experiments with data biases show that this would be non-trivial). Finally, further work will be required to extract a full mechanistic understanding of the rules learned with this framework, and investigate the data features informative for meta-learning the rules.

Conclusion To summarize, our attempt to flexibly learn rules directly from data in a simple scenario reveals that several different plasticity rules can explain the same data equally well, provided the generative model of plasticity is expressive enough. This suggests a shift in the way we think about

plasticity rules: not as individual plasticity rules, but rather families of rules with similar network-level function and potential mechanistic differences. And rather than studying one rule at a time, one would then attempt to characterize the properties which are conserved across equivalence classes (or universality classes) of rules. Predictions *shared* across many rules within a class are likely to be more robust than predictions idiosyncratic to a specific rule.

Acknowledgments and Disclosure of Funding

We thank Everton J. Agnes, Friedemann Zenke, Chaintanya Chintaluri, Auguste Schulz, Richard Gao and Julius Vetter for helpful discussions and feedback on the manuscript. This work was funded by the German Research Foundation (DFG; Germany's Excellence Strategy MLCoE – EXC number 2064/1 PN 390727645), the German Federal Ministry of Education and Research (BMBF; Tübingen AI Center, FKZ: 01IS18039A) and the European Reseach Council (ERC consolidator grant SYNAPSEEK).

References

Larry F Abbott and Sacha B Nelson. Synaptic plasticity: taming the beast. *Nature neuroscience*, 3 (11):1178–1183, 2000.

Ami Citri and Robert C. Malenka. Synaptic plasticity: Multiple forms, functions, and mechanisms. *Neuropsychopharmacology*, 33:18–41, 2008.

Jeffrey C. Magee and Christine Grienberger. Synaptic plasticity forms and functions. *Annual Review of Neuroscience*, 43(1):95–117, 2020.

Abigail Morrison, Ad Aertsen, and Markus Diesmann. Spike-timing-dependent plasticity in balanced random networks. *Neural computation*, 19(6):1437–1467, 2007.

Wulfram Gerstner, Richard Kempter, J. Leo van Hemmen, and Hermann Wagner. A neuronal learning rule for sub-millisecond temporal coding. *Nature*, 383:76–78, 1996.

Jean-Pascal Pfister and Wulfram Gerstner. Triplets of spikes in a model of spike timing-dependent plasticity. *Journal of Neuroscience*, 26(38):9673–9682, 2006.

Claudia Clopath, Lars Büsing, Eleni Vasilaki, and Wulfram Gerstner. Connectivity reflects coding: a model of voltage-based stdp with homeostasis. *Nature Neuroscience*, 13:344–352, 2010.

Michael Graupner and Nicolas Brunel. Calcium-based plasticity model explains sensitivity of synaptic changes to spike pattern, rate, and dendritic location. *Proceedings of the National Academy of Sciences*, 109(10):3991–3996, 2012.

Tim P. Vogels, Henning Sprekeler, Claudia Clopath, and Wulfram Gerstner. Inhibitory plasticity balances excitation and inhibition in sensory pathways and memory networks. *Science*, 334, 2011.

Friedemann Zenke, Everton J Agnes, and Wulfram Gerstner. Diverse synaptic plasticity mechanisms orchestrated to form and retrieve memories in spiking neural networks. *Nature Communications*, 6, 2015.

Ashok Litwin-Kumar and Brent Doiron. Formation and maintenance of neuronal assemblies through synaptic plasticity. *Nature Communications*, 5, 2014.

James J Jun, Nicholas A Steinmetz, Joshua H Siegle, Daniel J Denman, Marius Bauza, Brian Barbarits, Albert K Lee, Costas A Anastassiou, Alexandru Andrei, Çağatay Aydın, et al. Fully integrated silicon probes for high-density recording of neural activity. *Nature*, 551(7679):232–236, 2017.

Jeffrey P Nguyen, Frederick B Shipley, Ashley N Linder, George S Plummer, Mochi Liu, Sagar U Setru, Joshua W Shaevitz, and Andrew M Leifer. Whole-brain calcium imaging with cellular resolution in freely behaving caenorhabditis elegans. *Proceedings of the National Academy of Sciences*, 113(8):E1074–E1081, 2016.

Basile Confavreux, Friedemann Zenke, Everton J Agnes, Timothy Lillicrap, and Tim P Vogels. A meta-learning approach to (re) discover plasticity rules that carve a desired function into a neural network. *Advances in Neural Information Processing Systems 34 (NeurIPS)*, 2020.

Danil Tyulmankov, Guangyu Robert Yang, and L.F. Abbott. Meta-learning synaptic plasticity and memory addressing for continual familiarity detection. *Neuron*, 110(3):544–557.e8, 2022.

Jack Lindsey and Ashok Litwin-Kumar. Learning to learn with feedback and local plasticity. *Advances in Neural Information Processing Systems 34 (NeurIPS)*, 2020.

Jakob Jordan, Maximilian Schmidt, Walter Senn, and Mihai A Petrovici. Evolving interpretable plasticity for spiking networks. *eLife*, 10:e66273, 2021.

Luke Metz, Niru Maheswaranathan, Brian Cheung, and Jascha Sohl-Dickstein. Learning unsupervised learning rules. *arXiv*, 2018.

Sukbin Lim, Jillian L. McKee, Yali Woloszyn, Luke Amit, David J. Freedman, David L. Sheinberg, and Nicolas Brunel. Inferring learning rules from distribution of firing rates in cortical neurons. *Nature neuroscience*, 18:1804–1810, 2015.

Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *arXiv preprint arXiv:1406.2661*, 2014.

Erkki Oja. Simplified neuron model as a principal component analyzer. *Journal of mathematical biology*, 15(3):267–273, 1982.

Christopher M Bishop et al. *Neural networks for pattern recognition*. Oxford university press, 1995.

Wulfram Gerstner and Werner M Kistler. Mathematical formulations of hebbian learning. *Biological cybernetics*, 87(5):404–415, 2002.

Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

Abhishek Gupta, Benjamin Eysenbach, Chelsea Finn, and Sergey Levine. Unsupervised meta-learning for reinforcement learning, 2018.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations, 2018.

Liu Yang, Dongkun Zhang, and George Em Karniadakis. Physics-informed generative adversarial networks for stochastic differential equations, 2018.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans, 2016.

Akash Srivastava, Lazar Valkov, Chris Russell, Michael U Gutmann, and Charles Sutton. Veegan: Reducing mode collapse in gans using implicit variational learning. *arXiv preprint arXiv:1705.07761*, 2017.

Ferenc Huszár. How (not) to train your generative model: Scheduled sampling, likelihood, adversary?, 2015.

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 H. Wallach, H. Larochelle, A. Beygelzimer, F. dAlché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

Lukas Biewald. Experiment tracking with weights and biases, 2020.

Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. *CoRR*, abs/1802.05957, 2018.

Checklist

your paper or providing a brief inline description. For example:

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] In the discussion section of the main paper.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] We plan to release code after the review process is complete, since the code is not anonymizable in its current state.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] In Appendix B
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] In Appendix B
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] In Appendix B
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Additional results

Local MLP before and after training In order to verify that arbitrary learning rules do not reproduce the observed postsynaptic activity statistics, we compared the outputs of the parametrized rules (local MLP) before and after training. We found that the untrained MLP rule produces weight

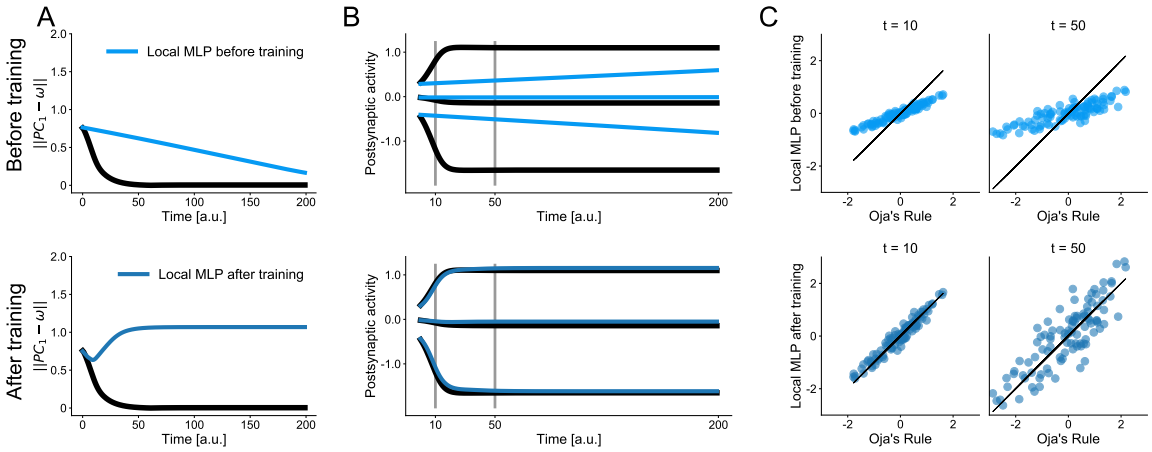

Figure A.1: Local MLP performance before (top) and after (bottom) GAN training. (A) Weight trajectories given by $||PC_1 - \omega||$ for Oja's rule (black) and learned rule (light / dark blue). (B) Postsynaptic activity traces for Oja's rule (black) and learned rule (light / dark blue). (C) Observed versus generated postsynaptic activity per timestep.

updates different from Oja's rule (Fig. A.1A). Furthermore, and in contrast with the learned rule, the rate network with the untrained MLP rule does not produce postsynaptic activity traces that resemble the traces from Oja's rule (Fig. A.1B), and does not capture their statistics (Fig. A.1C).

Multiple test datasets We simulated multiple datasets for the GANs, for training (D_{train}, containing 500 different datasets) and testing (D_{test}, containing 100 different datasets). For all of these datasets, the ground-truth synaptic plasticity rule was always Oja's rule (Eqn. 2), and $K = 100$ postsynaptic activity traces y_i were simulated using the linear feedforward network described in Sec. 2. Correspondingly, we sampled $K = 100$ different presynaptic activity samples x_j for each dataset in D_{train} or D_{test} from a multivariate Gaussian distribution with a different random correlation matrix per dataset (see Appendix Sec. B for more details). We chose one dataset from D_{test} on which to perform the reported analyses (Figs. 2-4). In this dataset, the learned rules reproduce the statistics of the observed postsynaptic activity.

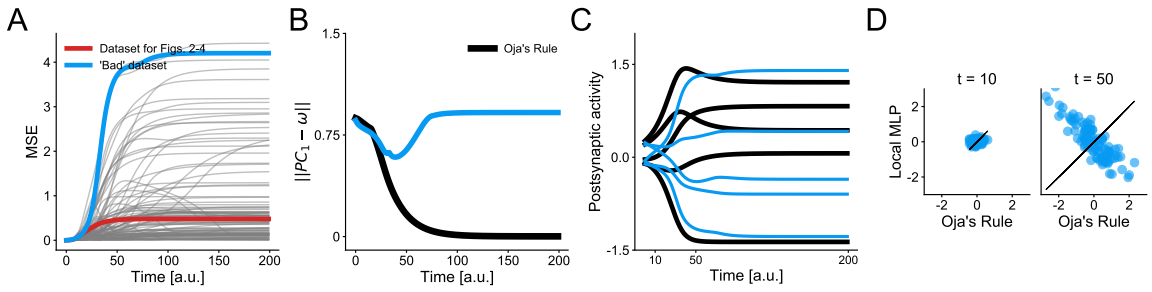

Figure A.2: Learned rules on multiple test datasets. (A) Mean squared error (MSE) of local MLP on multiple test datasets (grey). In blue, MSE for dataset in (B) and in red, dataset in Figs.2-4. (B) Weight trajectories for Oja's rule (black) and learned rule (blue). (C)Postsynaptic activity traces for Oja's rule (black) and learned rule (blue). (D) Observed versus generated postsynaptic activity per timestep.

In order to demonstrate that the chosen dataset is representative of the performance of the learned rule (local MLP), for each dataset in D_{test}, we computed the mean squared error (MSE) between the postsynaptic activity traces generated from the learned rule and from Oja's rule per timepoint t, averaged across $K = 100$ samples of x_j in each dataset. Note that for the $K = 100$ x_j samples and y_i traces, we also sampled the initial synaptic weights randomly $\omega_{ij}^0 \sim \mathcal{N}(0, 0.1^2)$, and used that same initialisation for both the observed and generated traces.

We found that most of the datasets in D_{test} had MSE traces that were consistently $<= 1$ (Fig. A.2A), and that the dataset we used for our experiments in Figs. 3-4 also had an MSE trace within this range. We also found that the datasets with MSE traces > 1 typically had high MSE values because the generated traces were anti-correlated with the observed traces (Fig. A.2C, D). However, this is also a feature of Oja's rule: the post-synaptic activity converges up to a sign to the projected presynaptic input (onto the first principal component of the input), and thus two simulations under Oja's rule may also have anti-correlated postsynaptic activity for the same initial synaptic weights.

Postsynaptic activity traces from different parametrizations of the synaptic plasticity rule We find that the postsynaptic activity traces from the differently parameterized learned rules (Sec.2 and Fig. 3) are qualitatively similar to the traces from Oja's rule: they all capture the transient change in activity at the early timesteps, and the convergence of the activity to a single value at later timesteps.

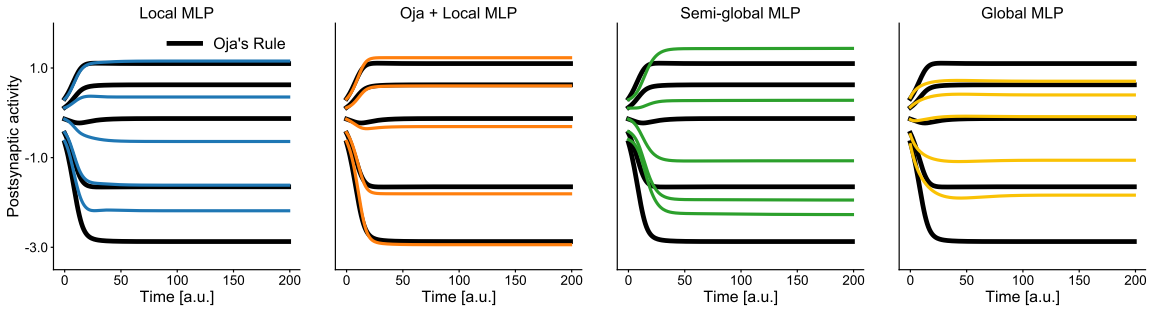

Figure A.3: Postsynaptic activity traces for local MLP (left, blue), Oja + local MLP (second from left, orange), semi-local MLP (second from right, green) and global MLP (right, yellow) compared to traces from Oja's rule (black)

MLP from Oja+local MLP does not contribute to the weight update We showed that the learned Oja + local MLP led to postsynaptic activity traces and weight updates that were virtually identical to the corresponding quantities with Oja's rule (Fig. 3). To show that the component 'local MLP' from 'Oja + local MLP' does not contribute significantly to either the weight update or the activity traces, we performed the same analysis as for the local MLP in Fig. 2. We found that the

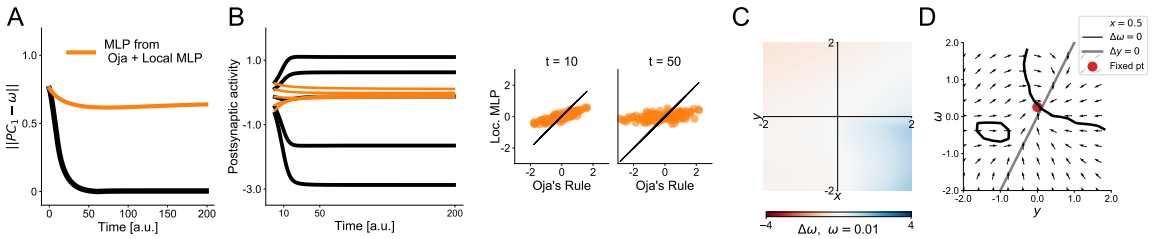

Figure A.4: Contribution of the component 'local MLP' for the learned rule Oja + local MLP. (A) Weight trajectory for MLP (orange) and Oja's rule (black), (B) Left: postsynaptic activity traces for MLP and Oja's rule. Right: observed versus generated activity per timestep. (C) Weight updates from MLP for a range of x_j and y_i values, with $\omega_{ij} = 0.01$. (D) Vector field of ω versus postsynaptic activity y for MLP, with $x = 0.5$.

synaptic weights (Fig. A.4A) and postsynaptic activity (Fig. A.4B) do not change much from their

initial value, under the local MLP. We also found that the weight updates from the MLP for a range of $x_j - y_i$ values are close to 0 (Fig. A.4C). Finally, the phase portrait for the MLP shows a stable fixed point close to 0 for ω and y (Fig. A.4D), indicating that the MLP does not change the fixed point for Oja's rule. However, the MLP perturbs the dynamics of the rule away from the fixed point (Fig. A.5). Note that we recover the phase portrait for Oja + local MLP (Fig. 3) simply by adding the weight and postsynaptic updates computed independently for the MLP and Oja's rule (Fig. A.5).

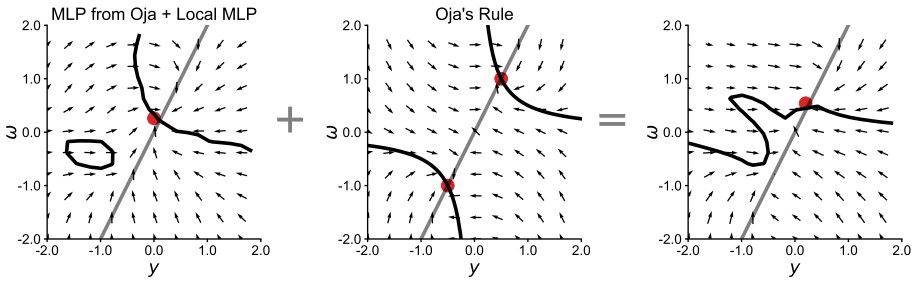

Figure A.5: $\Delta\omega_{ij}$ and Δy_i, computed independently for the component 'local MLP' from Oja + local MLP (left) and for the component 'Oja's rule' (middle), can be summed to recover the phase-portrait for Oja + local MLP.

Generalization for rules trained on the rate network with noisy postsynaptic activity or with 39 presynaptic neurons As a sanity check for the local MLP rules trained on data with noisy postsynaptic activity $y_i + \epsilon$ or 39 presynaptic neurons, we investigated whether these models were able to capture the statistics of held out datasets, that were within-distribution with respect to the training dataset. We found that the rule trained on noisy postsynaptic activity was able to capture the statistics of noisy postsynaptic activity from a held out dataset, while still producing weight trajectories dissimilar to Oja's rule (Fig. A.6A). This was also the case for the local MLP trained with data simulated with 39 presynaptic neurons (Fig. A.6B).

B Implementation details

We implemented the network model and GANs in the PyTorch framework [Paszke et al., 2019]. We used Weights and Biases Biewald [2020] to log different GAN training experiments. We trained all GANs on NVIDIA RTX 2080Ti GPUs, and training took on average 11 hours per GAN for 20k epochs.

Network model implementation

The network model was feedforward linear, with N presynaptic neurons and M postsynaptic neuron, and thus, $M \times N$ synaptic weights. The network received as input the presynaptic activity $\mathbf{x}$, a $K \times N$-dimensional vector: K samples from a N-D Gaussian distribution with a specified covariance matrix Σ i.e., $\mathbf{x} \sim \mathcal{N}(0, \Sigma)$. We specified the covariance matrix as follows:

- We constructed a $N \times N$ matrix $\mathbf{A}$, by sampling each element from a uniform distribution: $\mathbf{A}_{ij} \sim \mathcal{U}(0, 1)$
- We performed QR-decomposition on $\mathbf{A}$ to obtain an orthogonal matrix $\mathbf{Q}$, and ensured that the $\mathbf{Q}$ was positive definite by flipping the signs of its elements, if its determinant was negative.
- We fixed a diagonal matrix $\mathbf{D}$ (the values on the diagonal were different depending on the value of N).
- We computed the covariance matrix $\Sigma = \mathbf{Q}^T\mathbf{D}\mathbf{Q}$

We generated L different datasets, by sampling different $\mathbf{A}$ and thus constructing different Σ for each dataset, and correspondingly sampling K different postsynaptic activity vectors $\mathbf{x} \sim \mathcal{N}(0, \Sigma)$ for each Σ.

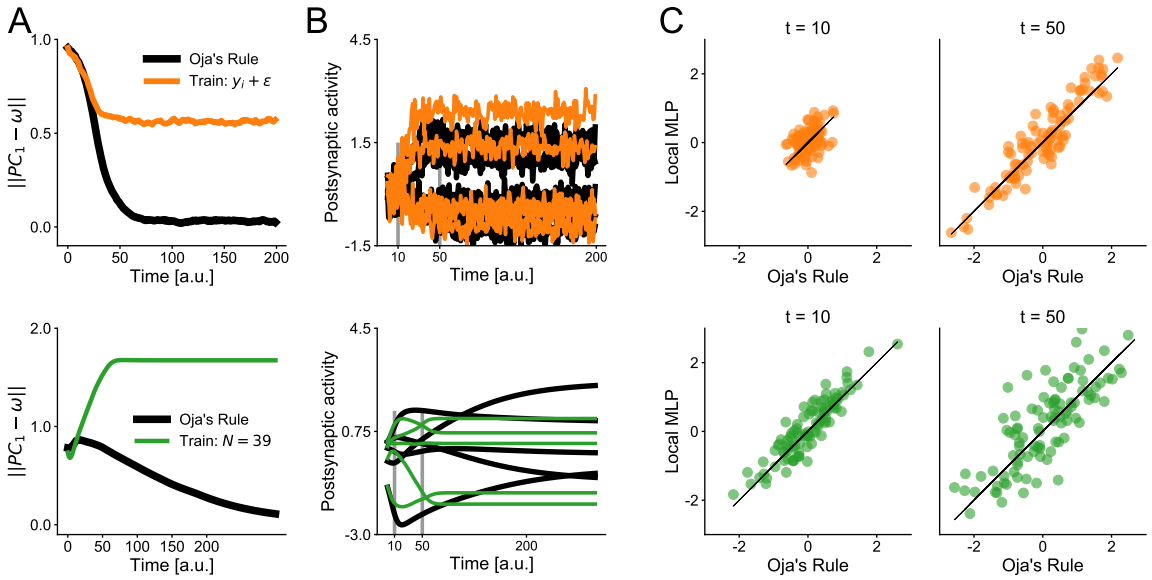

Figure A.6: Learned rules capture statistics of within-distribution data. (A) Weight trajectories. Top: local MLP trained with noisy postsynaptic activity (orange) and Oja's rule simulated with noisy postsynaptic activity (black). Bottom: local MLP trained with $N = 39$ presynaptic neurons (green) and Oja's rule simulated with 39 presynaptic neurons (black). (B) Postsynaptic activity traces for Oja's rule (black) and learned rule on noisy postsynaptic activity (top, orange), and with 39 presynaptic neurons (bottom, green). (C) Observed versus generated postsynaptic activity per timepoint.

The postsynaptic activity at time t y_1^t was updated using:

$$y_i^t = \sum_{j=1}^{N} \omega_{ij}^t x_j^t \tag{B.1}$$

where ω_{ij} is the synaptic weight between the ith postsynaptic and jth presynaptic neuron. We simulated postsynaptic activity for T timesteps i.e., there were T updates to the postsynaptic activity in one trace, for each of the K presynaptic activity samples, for all L datasets. We kept the presynaptic activity the same for all T timesteps i.e., $x_j^t = x_j^0 = x_j$.

We updated the synaptic weights ω_{ij} concurrently with the postsynaptic activity, using an update rule h, which we denote as h_θ when the update function is parametrized. Note that the weights were updated using implicit batch learning, i.e. we computed the update $\Delta\omega_{ij} = h(\cdot)$ by averaging over the K different x_j and y_i^t values at teach timestep:

$$\omega_{ij}^{t+1} = \omega_{ij}^t + \frac{\eta}{K} \sum_{k=1}^{K} h(x_j^{(k)}, (y_i^{(k)})^t, \omega_{ij}^t) \tag{B.2}$$

where η is the learning rate. We thus had one synaptic weight trajectory corresponding to batch learning over K different pre- and post-synaptic activity traces for T timesteps. At the start of a simulation over K samples, we set the postsynaptic activity at $t = 0$ to 0, and initialised the synaptic weights randomly ($\omega_{ij}^0 \sim \mathcal{N}(0, 0.1^2)$).

For all experiments described in the main paper, we set $T = 200$, $M = 1$, $K = 100$, and $\eta = 0.1$.

For the proof of principle, and training the local MLP in Fig. 2, and for the test data in Fig. 2-4, we set $N = 3$.

For the noisy postsynaptic activity training data in Fig. 4, we set $N = 3$, and modified Eqn B.1 as follows:

$$y_i^t = \sum_{j=1}^{N} \omega_{ij}^t x_j^t + \epsilon, \ \epsilon \in \mathbb{R}^M, \ \epsilon \sim \mathcal{N}(0, 0.25^2) \tag{B.3}$$

For the training data with 39 presynaptic neurons in Fig. 4, we set $N = 39$.

Synaptic update rules

For all training and test data, the groundtruth rule h was Oja's rule (Eqn. 2)

The parametrized update rules h_θ had the following architecture:

1. **Constrained rule**: $h_\theta(\omega_{ij}^t, x_j, y_i^t) = y_i^t(\theta_1 x_j + \theta_2 y_i^t \omega_{ij}^t)$.
   The parameters of this rule were the scalars θ_1 amd θ_2
2. **Local MLP**:
   This was an MLP with 3 fully-connected layers with hidden units $[3, 2, 1]$ and added bias. Each of the fully-connected layers was followed by a non-linearity: [SiLU, LeakyReLU with slope 0.2, LeakyReLU with slope 0.2]. This MLP was local: it took a 3-dimensional input consisting of one value of the presynaptic activity for one neuron e.g. j, the postsynaptic activity of one neuron i, and the synaptic weight connecting the two ω_{ij}.
3. **Oja + local MLP**:
   The MLP for this rule had the exact same architecture as for Local MLP. We then added the output of Oja's rule to the output of the MLP.
4. **Semi-global MLP**:
   This MLP had the same architecture as for Local MLP, except that it took a 5D input. The two additional dimensions compared to local MLP were the mean presynaptic activity from N presynaptic neurons, and the mean synaptic weight across $M \times N$ synaptic weights.
5. **Global MLP**:
   This MLP had 3-fully connected layers with hidden units $[16, 16, M \times N]$ and added bias, followed by a final layer that reshaped the output of the last fully-connected layer into a $K \times N \times M$ tensor. Each of the fully-connected layers was followed by a LeakyReLU non-linearity with slope 0.2. This MLP was global: it took the activity of all pre- and postsynaptic neurons in the network as well as all synaptic weights as input, and computed an update for all synaptic weights in a single forward pass.

Discriminator architecture

For the learned rules, the discriminator architecture was as follows:

1. For the constrained rule: 2 convolutional layer with input channels , output channels and kernel size = ($T = 10$, 5, 10) and (5, 3, 3). Each convolutional layer was followed by a 1D MaxPool layer with kernel size 2, and stride 1. These layers were followed by a fully connected layer with 1 hidden unit. Each layer was also followed by a LeakyReLU nonlinearity with slop 0.2. The final fully connected layer was followed by a sigmoid.
2. For all other parametrized rules: 2 convolutional layers with input channels , output channels and kernel size = (M, 5, 10) and (5, 1, 10). These layers were followed by 2 fully connected layers with 64 and 1 hidden unit respectively. All layers were followed by a LeakyReLU nonlinearity with slope 0.2. However, the final fully connected layer was followed by a sigmoid. For these discriminators, we also used spectral normalisation Miyato et al. [2018] to stabilize training.

Training details

: To train all GANs, we simulated $L = 500$ datasets, one of which was held out for validation to check if training had converged. The batchsize for all GANs was set to 1 i.e., we passed the GANs one dataset per update of the generator and discriminator networks. We also use gradient norm clipping to stabilise training. From the network with $N = 3, \epsilon = 0$, we additionally generated $L' = 100$ test datasets, on which we performed our post-training analsis. For the networks with 39 presynaptic neurons and with noisy postsynaptic activity, we generated $L' = 20$ test datasets.

For the minimial network, we trained the parameters for 5k epochs, where each epoch consisted of 10 discriminator and 1 generator update. Note that we used only $T = 10$ timesteps of the simulated

data to train these networks. We used the Adam optimiser with learning rate 0.0001, $\beta_1 = 0.9$ and $\beta_2 = 0.999$.

For the local MLP, we trained for 27k epochs with 10 discriminator and 10 generator updates in each epoch. We clipped the gradients of both networks to be below 0.01, and used all $T = 200$ timesteps of the postsynaptic activity.

For the semi-local MLP, we trained for 40k epochs with 2 discriminator and 2 generator updates in each epoch. We clipped the gradients of both networks to be below 0.001, and used all $T = 200$ timesteps of the postsynaptic activity.

For the global MLP, we trained for 31k epochs with 5 discriminator and 5 generator updates in each epoch. We clipped the gradients of both networks to be below 0.001, and used all $T = 200$ timesteps of the postsynaptic activity.

For the local MLP on noisy data, we trained for 20k epochs with 10 discriminator and 5 generator updates in each epoch. We clipped the gradients of both networks to be below 0.01, and used all $T = 200$ timesteps of the postsynaptic activity.

For the local MLP on data with 39 presynaptic neurons, we trained for 20k epochs with 10 discriminator and 10 generator updates in each epoch. We clipped the gradients of both networks to be below 0.001, and used all $T = 200$ timesteps of the postsynaptic activity.