# Score Modeling for Simulation-based Inference

**Tomas Geffner**[*]
University of Massachusetts

**George Papamakarios**
DeepMind

**Andriy Mnih**
DeepMind

## Abstract

Neural Posterior Estimation methods for simulation-based inference can be ill-suited for dealing with posterior distributions obtained by conditioning on multiple observations, as they may require a large number of simulator calls to yield accurate approximations. Neural Likelihood Estimation methods can naturally handle multiple observations, but require a separate inference step, which may affect their efficiency and performance. We introduce a new method for simulation-based inference that enjoys the benefits of both approaches. We propose to model the scores for the posterior distributions induced by individual observations, and introduce a sampling algorithm that combines the learned scores to approximately sample from the target efficiently.

## 1 Introduction

Mechanistic simulators have been developed in a wide range of scientific domains [3]. Often these simulators act as a black box: given a set of parameters, the simulator can be sampled, but the distribution over the outputs–the likelihood–cannot be evaluated, rendering typical inference algorithms inapplicable. Simulation-based inference (SBI) methods provide a way to perform inference with these models [1, 3]. Given a prior over parameters $p(\theta)$ and a simulator for the likelihood $p(x|\theta)$, the goal of SBI is to approximate the posterior $p(\theta|x_1^o, \dots, x_n^o)$ for any set of i.i.d. observations $\{x_1^o, \dots, x_n^o\}$. Most SBI methods work by running the simulator to generate samples $x \sim p(x|\theta)$ for different parameters $\theta$, and using the resulting samples to build an approximation of the posterior. Since many domains involve expensive simulators, recent work has focused on developing algorithms that yield good approximations using a limited budget of simulator calls.

Recent work introduced Neural Posterior Estimation (NPE) methods [20, 17, 8, 2], which use samples $(\theta, x_1, \dots, x_n) \sim p(\theta) \prod_{j=1}^n p(x_j|\theta)$ to train a conditional neural density estimator $q_\psi(\theta|x_1, \dots, x_n)$ with parameters $\psi$, often a normalizing flow [26, 33, 37], via maximum likelihood. After training, the estimator provides an amortized approximation to $p(\theta|x_1^o, \dots, x_n^o)$ for any set of observations $\{x_1^o, \dots, x_n^o\}$ of size $n$.[1] The drawback of NPE methods is that each training sample requires $n$ simulator calls, so building a training set of size $M$ requires running the simulator $nM$ times. This may be problematic in scenarios where $n$ is large and calls to the simulator are expensive.

Neural Likelihood Estimation (NLE) methods [21, 39, 15] are a natural alternative for cases where $n > 1$. These methods learn a surrogate likelihood $q_\psi(x|\theta)$ (or a likelihood ratio [22, 4, 10]) using samples $(\theta, x) \sim \tilde{p}(\theta)p(x|\theta)$, where $\tilde{p}(\theta)$ is a proposal distribution, which in the simplest case can default to the prior $p(\theta)$. Then, given a set of observations $\{x_1^o, \dots, x_n^o\}$, inference is carried out on the approximate unnormalized target $p(\theta) \prod_j^n q_\psi(x_j^o|\theta)$ by standard methods, typically MCMC [21, 15] or variational inference [38, 7]. While these methods can handle arbitrary sets of observations at inference time without re-training the surrogate, they do not approximate the posterior directly, and thus require the inference step to be repeated for each set of observations of interest. Moreover, their

[*]Work done during an internship at DeepMind.

[1]NPE methods can handle sets of observations with varying cardinality as well [23], by training the density estimator using samples $(\theta^i, x_1^i, \dots, x_{n_i}^i) \sim p(\theta) \prod_j^{n_i} p(x_j|\theta)$ of varying size $n_i$, and conditioning it not only on the samples $x_1, \dots, x_{n_i}$ but also on the cardinality of the set $n_i$. We give details in Appendix B.

NeurIPS 2022 Workshop on Score-Based Methods.

performance depends on the performance of the underlying generic inference methods, which tend to struggle e.g. with multimodal distributions.

Our goal is to develop a method that enjoys the benefits of both types of approaches while avoiding their drawbacks—a method that approximates the posterior directly, is trained on samples $(\theta, x) \sim p(\theta)p(x|\theta)$, and is able to naturally handle a varying number of observations at test time. We propose such an approach that relies on score-based modeling [30, 31, 11, 32]. Simply put, we train a single conditional score network to approximate the score of (diffused versions of) $p(\theta|x)$ for any $x$, and propose an algorithm that uses the trained network to approximately sample the posterior $p(\theta|x_1^o, \dots, x_n^o)$ for *any* set of observations $\{x_1^o, \dots, x_n^o\}$. Our method satisfies the three desiderata outlined above: it directly approximates the posterior, learns form samples $(\theta, x) \sim p(\theta)p(x|\theta)$ produced with a single call to the simulator, and provides a sampling algorithm that can handle arbitrary sets of observations without re-training.

### 1.1 Conditional score-based generative modeling

The goal of conditional generative modeling is to learn an approximation of a target distribution $p(\theta|c)$ for some conditioning variable $c$ given samples $(\theta, c) \sim p(\theta, c)$, which is exactly the problem SBI methods need to solve. Methods based on score modeling have shown impressive performance for this task [31, 5, 12, 25, 28, 29]. They define a sequence of conditional densities $p_0(\theta|c), \dots, p_T(\theta|c)$ by diffusing the target $p(\theta|c)$ with Gaussian kernels of increasing levels of noise, learn the scores of each density in the sequence using denoising score matching [13, 35], and use Langevin dynamics [27, 36] with the learned scores to approximately sample from the target distribution.

Specifically, for the noise levels $0 = \gamma_T < \gamma_{T-1} < \dots < \gamma_1 < 1$ and the corresponding Gaussian diffusion kernels $p_t(\theta|\theta') = \mathcal{N}(\theta|\sqrt{\gamma_t}\,\theta', (1-\gamma_t)I)$, the sequence of densities is defined as

$$p_0(\theta|c) = p(\theta|c) \quad \text{and} \quad p_t(\theta|c) = \int d\theta'\, p(\theta'|c) p_t(\theta|\theta') \quad \text{for } t = 1, \dots, T. \tag{1}$$

Since $\gamma_T = 0$, this sequence gradually bridges between the initial tractable reference $\mathcal{N}(\theta|0, I) = p_T(\theta)$ and the target $p(\theta|c) = p_0(\theta|c)$. Score-based methods train a score network $s_\psi(\theta, t, c)$ parameterized by $\psi$ to approximate the scores of these densities, $\nabla_\theta \log p_t(\theta|c)$. As only samples from the target are available, this is done via denoising score matching [13, 35], minimizing

$$\mathcal{L}_{\text{DSM}}(\psi) = \textstyle\sum_{t=1}^{T-1} \mathbb{E}_{p(\theta', c) p_t(\theta|\theta')} \left[ \left\| s_\psi(\theta, t, c) - \nabla_\theta \log p_t(\theta|\theta') \right\|^2 \right]. \tag{2}$$

Finally, the score network is used to approximately sample the target using annealed Langevin dynamics, as shown in Algorithm 1.

**Algorithm 1** Approximate sampling with learned scores

**Input:** Score network $s_\psi(\theta, t, c) \approx \nabla_\theta \log p_t(\theta|c)$, reference distribution $p_T(\theta)$
**Input:** Conditioning variable $c$, number of Langevin steps $L$, Langevin step sizes $\delta_t$

$\theta \sim p_T(\theta)$ ▷ Sample reference
**for** $t = T-1, T-2, \dots, 1$ **do**
  **for** $s = 1, 2, \dots, L$ **do**
    $\theta \leftarrow \theta + \frac{\delta_t}{2} s_\psi(\theta, t, c) + \sqrt{\delta_t}\,\eta_{ts}$ $\quad [\eta_{ts} \sim \mathcal{N}(0, I)]$ ▷ Unadjusted Langevin step
**return** $\theta$

## 2 Score-based Neural Posterior Estimation

This section presents our approach for SBI using score modeling. Our goal is to develop a method that can be trained using parameter/single-observation pairs $(\theta, x) \sim p(\theta)p(x|\theta)$, and that can be used at test time to approximate $p(\theta|x_1^o, \dots, x_n^o)$ for arbitrary sets of observations $\{x_1^o, \dots, x_n^o\}$ with any cardinality $n$. As we explain in Appendix A, a naive application of conditional score based modeling fails to satisfy our desiderata. Instead, we propose an alternative approach based on score modeling, involving different choices for the bridging densities and the reference distribution.

Our method is based on the observation that $p(\theta|x_1, ..., x_n) \propto p(\theta)^{1-n} \prod_{j=1}^n p(\theta|x_j)$ (see Appendix A). Using this factorization, we propose the sequence of densities

$$p_t(\theta|x_1, \dots, x_n) \propto \left(p(\theta)^{1-n}\right)^{\frac{T-t}{T}} \prod_{j=1}^n p_t(\theta|x_j) \qquad \text{for } \; t = 0, \dots, T, \tag{3}$$

where $p_t(\theta|x_j)$ is defined in Eq. (1), taking $c = x_j$. This construction has four key properties. First, the distribution for $t = 0$ recovers the target $p(\theta|x_1, ..., x_n)$. Second, the distribution for $t = T$ is a tractable Gaussian $p_T(\theta|x_1, \dots, x_n) = p_T(\theta) = \mathcal{N}(\theta|0, \frac{1}{n}I)$,[2] and thus can be used as a reference for the process. Third, the score of the resulting densities can be decomposed in terms of the score of the prior (available exactly) and the scores of $p_t(\theta|x_j)$ as

$$\nabla_\theta \log p_t(\theta|x_1, \dots, x_n) = \frac{(1-n)(T-t)}{T} \nabla_\theta \log p(\theta) + \sum_{j=1}^{n} \nabla_\theta \log p_t(\theta|x_j). \tag{4}$$

And fourth, the scores $\nabla_\theta \log p_t(\theta|x_j)$ can all be approximated using a ***single*** score network $s_\psi(\theta, t, x)$ trained via denoising score matching using samples $(\theta, x) \sim p(\theta)p(x|\theta)$, as explained in Section 1.1.

After training, given an arbitrary set of observations $\{x_1^o, \dots, x_n^o\}$ we can approximately sample the target $p(\theta|x_1^o, \dots, x_n^o)$ by running Algorithm 1 with the reference distribution $p_T(\theta) = \mathcal{N}(\theta|0, \frac{1}{n}I)$, conditioning variable $c = \{x_1^o, \dots, x_n^o\}$, and the approximate score given by

$$s_\psi(\theta, t, c) = \frac{(1-n)(T-t)}{T} \nabla_\theta \log p(\theta) + \sum_{j=1}^{n} s_\psi(\theta, t, x_j^o). \tag{5}$$

It is straightforward to verify that our approach satisfies our original desiderata: the score network $s_\psi(\theta, t, x)$ is trained using samples $(\theta, x) \sim p(\theta)p(x|\theta)$, and the sampling algorithm can be used with any set of observations $\{x_1^o, \dots, x_n^o\}$, as it relies on Langevin dynamics with Eq. (5).

### 2.1 Alternative sampling approach

The sampling process described in Algorithm 1 requires choosing step-sizes $\delta_t$, the number of steps $L$ per noise level, and has complexity $\mathcal{O}(LT)$. This section introduces a different method to approximately sample the target $p(\theta|x_1, \dots, x_n)$, which does not use Langevin dynamics and runs in $\mathcal{O}(T)$ steps. The approach is based on the formulation by Sohl-Dickstein et al. [30] to use diffusion models to approximately sample from a product of distributions. The final method involves sampling $T$ Gaussian transitions with means and variances computed using the learned score network. We describe the method in Algorithm 2, and give its derivation in Appendix C.

**Algorithm 2** Approximately sampling without unadjusted Langevin dynamics

**Input:** Score network $s_\psi(\theta, t, x) \approx \nabla_\theta \log p_t(\theta|x)$, reference distribution $p_T(\theta)$
**Input:** Conditioning variables $x_1, \dots, x_n$, noise levels $\gamma_1, \dots, \gamma_T$
  Define $\alpha_1 = \gamma_1$, $\alpha_t = \gamma_t / \gamma_{t-1}$, and $\beta_t = 1 - \alpha_t$, for $t = 1, \dots, T-1$
  $\theta \sim p_T(\theta)$ ▷ Sample reference
  **for** $t = T-1, T-2, \dots, 1$ **do**
    $\mu_{jt} = \frac{1}{\sqrt{\alpha_t}}(\theta + (1-\alpha_t)s_\psi(\theta, t, x_j))$ for $j = 1, \dots, n$ ▷ Compute auxiliary variables
    $\sigma_t^2 = \frac{\beta_t}{n - \alpha_t(n-1)}$, $\mu_t = \frac{\sum_j \mu_{jt} - (n-1)\sqrt{\alpha_t}\theta}{n - \alpha_t(n-1)} + \frac{\sigma_t^2(1-n)(T-t)}{T} \nabla_\theta \log p(\theta)$
    ▷ Set transition mean and variance
    $\theta \sim \mathcal{N}(\theta|\mu_t, \sigma_t^2 I)$ ▷ Sample transition
  **return** $\theta$

## 3 Empirical Evaluation

This section presents an empirical evaluation on two problems commonly used to evaluate SBI methods [16]. One involves a "simulator" consisting of a Gaussian prior and likelihood, $p(\theta) = \mathcal{N}(\theta|0, I)$ and $p(x|\theta) = \mathcal{N}(x|\theta, \Sigma)$ (we set $\Sigma$ to a diagonal matrix with elements increasing linearly from 0.6 to 1.4), while the other uses a Gaussian prior and a mixture-of-Gaussians likelihood, $p(\theta) = \mathcal{N}(\theta|0, I)$ and $p(x|\theta) = 0.5\mathcal{N}(x|\theta, 2.25\Sigma) + 0.5\mathcal{N}(x|\theta, \frac{1}{9}\Sigma)$. In both cases we set $\theta, x \in \mathbb{R}^{10}$.

We compare our approach, called Score NPE, to NPE using a normalizing flow with four Real NVP layers [6] (details in Appendix B). We compare the methods' performance when trained on datasets of different sizes, corresponding to different budgets of simulator calls $B \in \{10^3, 3 \cdot 10^3, 10^4, 3 \cdot 10^4\}$. In all cases we use 20% of the training data as a validation set for early stopping, and train for a maximum of 20k epochs. We train all methods using Adam [14] with a learning rate of $10^{-4}$.

[2] Since the prior term vanishes and $p_T(\theta|x_j) = \mathcal{N}(\theta|0, I)$ for all $j$.

After training we generate a set of observations by drawing $\theta \sim p(\theta)$ and $x_1^o, \dots, x_8^o \sim_{\text{iid}} p(x|\theta)$, and report the squared MMD [9] between the true posterior and the approximation returned by each method for subsets of $\{x_1^o, \dots, x_8^o\}$ of different size. Figure 1 shows average results over 40 random seeds. We observe that both methods perform similarly for the simpler Gaussian-Gaussian model, but that Score NPE outperforms the flow baseline for the model with the mixture-of-Gaussians likelihood.

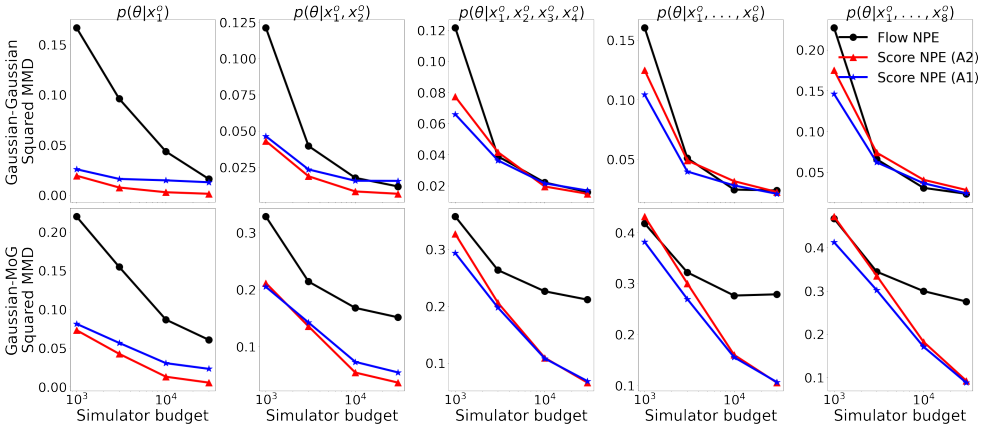


Figure 1: Squared MMD between the true posterior and approximation returned by different methods. (A1) and (A2) refer to using Algorithms 1 and 2 for sampling (details for step size and other parameters are in Appendix B). The MMD is computed using a Gaussian kernel with scale determined by the median heuristic [24]. Samples from the true posterior were obtained with HMC [19].

**Multimodal posterior.** We also consider a two-dimensional example with a multimodal posterior, with the prior and likelihood given by $p(\theta) = \mathcal{N}(\theta|0, I)$ and $p(x|\theta) = 0.5\mathcal{N}(x|\theta, 0.5I) + 0.5\mathcal{N}(x|-\theta, 0.5I)$. We train each method using a budget of $10^4$ simulator calls. After training we sample $\theta \sim p(\theta)$ and $x_1^o, \dots, x_5^o \sim_{\text{iid}} p(x|\theta)$, and use each method to generate samples from the approximate posterior obtained by conditioning on subsets of $\{x_1^o, \dots, x_5^o\}$ of different size. Results are shown in Fig. 2, where it can be observed that our method is able to capture both modes well for all subset sizes of observations despite only being trained on parameters/single-observation pairs.

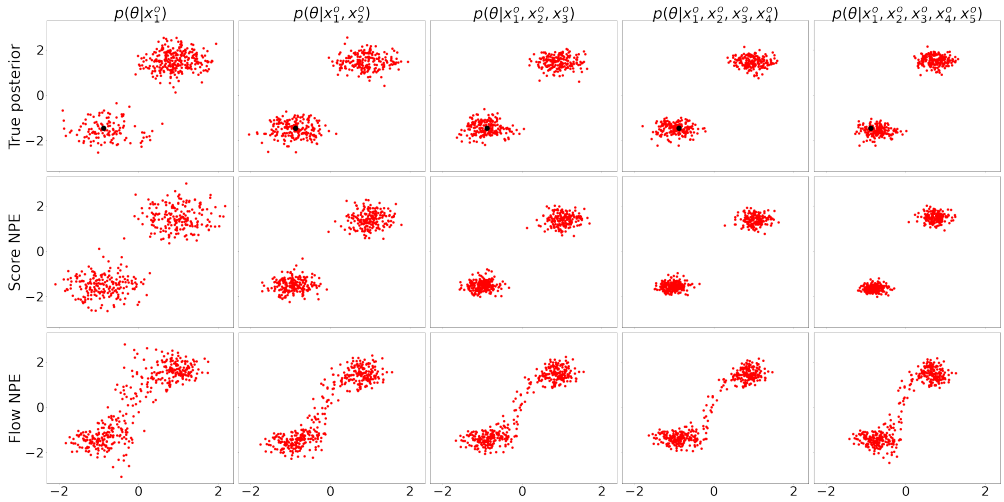


Figure 2: Posteriors for the multimodal example. True parameters $\theta$ used to generate $x_1^o, \dots, x_5^o$ are shown in black in the first row. Score NPE samples were obtained using Algorithm 2.

## Acknowledgments and Disclosure of Funding

We thank Francisco Ruiz for useful comments and suggestions.

## References

[1] Mark A Beaumont. Approximate Bayesian computation. *Annual review of statistics and its application*, 6:379–403, 2019.

[2] Jeffrey Chan, Valerio Perrone, Jeffrey Spence, Paul Jenkins, Sara Mathieson, and Yun Song. A likelihood-free inference framework for population genetic data using exchangeable neural networks. *Advances in Neural Information Processing Systems*, 31, 2018.

[3] 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.

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

[5] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat GANs on image synthesis. *Advances in Neural Information Processing Systems*, 34:8780–8794, 2021.

[6] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using Real NVP. *arXiv preprint arXiv:1605.08803*, 2016.

[7] Manuel Glöckler, Michael Deistler, and Jakob H Macke. Variational methods for simulation-based inference. *arXiv preprint arXiv:2203.04176*, 2022.

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

[9] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012.

[10] Joeri Hermans, Volodimir Begy, and Gilles Louppe. Likelihood-free MCMC with amortized approximate ratio estimators. In *International Conference on Machine Learning*, pages 4239–4248. PMLR, 2020.

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

[12] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

[13] Aapo Hyvärinen and Peter Dayan. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4), 2005.

[14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[15] Jan-Matthis Lueckmann, Giacomo Bassetto, Theofanis Karaletsos, and Jakob H Macke. Likelihood-free inference with emulator networks. In *Symposium on Advances in Approximate Bayesian Inference*, pages 32–53. PMLR, 2019.

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

[17] Jan-Matthis Lueckmann, Pedro J Goncalves, Giacomo Bassetto, Kaan Öcal, Marcel Nonnenmacher, and Jakob H Macke. Flexible statistical inference for mechanistic models of neural dynamics. *Advances in Neural Information Processing Systems*, 30, 2017.

[18] Calvin Luo. Understanding diffusion models: A unified perspective. *arXiv preprint arXiv:2208.11970*, 2022.

[19] Radford M Neal. MCMC using Hamiltonian dynamics. *Handbook of Markov chain Monte Carlo*, 2(11):2, 2011.

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

[21] George Papamakarios, David Sterratt, and Iain Murray. Sequential neural likelihood: Fast likelihood-free inference with autoregressive flows. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 837–848. PMLR, 2019.

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

[23] 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.

[24] Aaditya Ramdas, Sashank Jakkam Reddi, Barnabás Póczos, Aarti Singh, and Larry Wasserman. On the decreasing power of kernel and distance based nonparametric hypothesis tests in high dimensions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

[25] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with CLIP latents. *arXiv preprint arXiv:2204.06125*, 2022.

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

[27] Gareth O. Roberts and Richard L. Tweedie. Exponential convergence of Langevin distributions and their discrete approximations. *Bernoulli*, 2(4):341–363, 1996.

[28] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S Sara Mahdavi, Rapha Gontijo Lopes, et al. Photorealistic text-to-image diffusion models with deep language understanding. *arXiv preprint arXiv:2205.11487*, 2022.

[29] Yuyang Shi, Valentin De Bortoli, George Deligiannidis, and Arnaud Doucet. Conditional simulation using diffusion Schrödinger bridges. *arXiv preprint arXiv:2202.13460*, 2022.

[30] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning*, pages 2256–2265. PMLR, 2015.

[31] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in Neural Information Processing Systems*, 32, 2019.

[32] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

[33] Esteban G Tabak and Cristina V Turner. A family of nonparametric density estimation algorithms. *Communications on Pure and Applied Mathematics*, 66(2):145–164, 2013.

[34] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

[35] Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

[36] Max Welling and Yee W Teh. Bayesian learning via stochastic gradient Langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning*, pages 681–688. Citeseer, 2011.

[37] Christina Winkler, Daniel Worrall, Emiel Hoogeboom, and Max Welling. Learning likelihoods with conditional normalizing flows. *arXiv preprint arXiv:1912.00042*, 2019.

[38] Samuel Wiqvist, Jes Frellsen, and Umberto Picchini. Sequential neural posterior and likelihood approximation. *arXiv preprint arXiv:2102.06522*, 2021.

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

# A Failure of direct application of conditional score modeling

The target distribution is given by $p(\theta|x_1, \dots, x_n)$. A direct application of conditional score modeling yields the sequence of densities

$$\begin{aligned} p_0(\theta|x_1, \dots, x_n) &= p(\theta|x_1, \dots, x_n) \\ p_t(\theta|x_1, \dots, x_n) &= \int d\theta' \, p(\theta'|x_1, \dots, x_n) p_t(\theta|\theta') \qquad \text{for } \; t = 1, \dots, T. \end{aligned} \tag{6}$$

It can be seen from the equation above that the score $\nabla_\theta \log p_t(\theta|x_1, \dots, x_n)$ does not factorize in terms of the single-observation scores $\nabla_\theta \log p_t(\theta|x_j)$, meaning that the corresponding score network would have to be trained using samples $(\theta, x_1, \dots, x_n) \sim \prod_j^n p(x_j|\theta)$, obtained by calling the simulator $n$ times for a single sample $\theta$. As mentioned in Section 1 this is one of the drawbacks of NPE methods that we seek to avoid.

### A.1 Derivation of posterior factorization

The factorization for the posterior distribution $p(\theta|x_1, \dots, x_n)$ is obtained applying Bayes rule twice:

$$p(\theta|x_1, \dots, x_n) \propto p(\theta) p(x_1, \dots, x_n|\theta) \qquad \text{(Bayes rule)} \tag{7}$$

$$= p(\theta) \prod_{j=1}^{n} p(x_j|\theta) \tag{8}$$

$$\propto p(\theta) \prod_{j=1}^{n} \frac{p(\theta|x_j)}{p(\theta)} \qquad \text{(Bayes rule)} \tag{9}$$

$$= p(\theta)^{1-n} \prod_{j=1}^{n} p(\theta|x_j). \tag{10}$$

# B Details for empirical evaluation

### B.1 Score NPE

Our implementation of the score network $s_\psi(\theta, t, x)$ has three blocks:

- An MLP with 3 hidden layers that takes $\theta$ as input and outputs an embedding $\theta_{\text{emb}}$,
- An MLP with 3 hidden layers that takes $x$ as input and outputs an embedding $x_{\text{emb}}$,
- An MLP with 3 hidden layers that takes $[\theta_{\text{emb}}, x_{\text{emb}}, t_{\text{emb}}]$ as input, where $t_{\text{emb}}$ is a positional embedding obtained as described by Vaswani et al. [34], and outputs the estimate for the score. (We parameterize the score in terms of the noise variables $\epsilon$ [18].)

All MLPs use residual connections throughout.

Running Algorithm 1 to generate samples using the learned score network requires choosing step sizes $\delta_t$ and the number of Langevin steps $L$ for each noise level $\gamma_t$. We use $L = 50$ and $\delta_t = 0.05 \frac{1-\alpha_t}{\sqrt{\alpha_t}}$, where $\alpha_1 = \gamma_1$ and $\alpha_t = \frac{\gamma_t}{\gamma_{t-1}}$ for $t = 2, \dots, T-1$. For all our experiments we use $T = 400$.

### B.2 Flow NPE

We use an implementation of NPE methods based on flows able to handle sets of observations of any size $n \in \{1, 2, \dots n_{\max}\}$. The flow can be expressed as $q_\psi(\theta|x_1, \dots, x_n, n)$. Following Chen et al. [2] and Radev et al. [23, §2.4], we use an exchangeable neural network to process the observations $x_1, \dots, x_n$. Specifically, we use an MLP with 3 hidden layers to generate an embedding $x_{ej}$ for each observation $x_j$. We then compute the mean embedding across observations $\bar{x}_e = \frac{1}{n}\sum_j^n x_{ej}$, which we use as input for the conditional flow. Finally, we model the flow $q_\psi(\theta|x_1, \dots, x_n, n) = q_\psi(\theta|\bar{x}_e, n_e)$, where $n_e$ is an untrained embedding for the number of observations $n$. For the flow we use 4 Real NVP layers [6], each one consisting on MLPs with three hidden layers. As for the Score NPE method, we use residual connections throughout.

We train the flow via maximum likelihood using samples $(n, \theta, x_1, \dots, x_n) \sim \text{Unif}(n|\min = 1, \max = 10)p(\theta)\prod_j^n p(x_j|\theta)$. At test time, this architecture can handle sets of observations of any size $n \in \{1, 2, \dots, 10\}$.

## C Alternative sampling method without unadjusted Langevin dynamics

This section gives the derivation for the sampling method shown in Algorithm 2. In short, the derivation uses the formulation of score-based methods as diffusions, and has 3 main steps: (1) using the scores of $p_t(\theta|x)$ to compute the Gaussian transition kernels of the corresponding diffusion process [18]; (2) composing $n$ Gaussian transitions corresponding to the $n$ diffusions of $p_t(\theta|x_1), \dots, p_t(\theta|x_n)$ (this is based on Sohl-Dickstein et al. [30]); and (3) adding a correction for the prior term (also based on Sohl-Dickstein et al. [30]). We note that steps 2 and 3 require some approximations. Despite this, our empirical evaluation indicates that the method works well in practice. We believe a thorough analysis of these approximations would be useful in understanding when the sampling method from Appendix C can be expected to work. For clarity, we use [A] to indicate when the approximations are introduced/used.

**Connection between score-based methods and diffusion models** We begin by noting that score-based methods can be equivalently formulated as diffusion models, where the mean of Gaussian transitions that act as denoising steps are learned instead of the scores. Specifically, letting $\alpha_1 = \gamma_1$, $\alpha_t = \gamma_t/\gamma_{t-1}$, and $\beta_t = 1 - \alpha_t$, for $t = 1, \dots, T-1$, the learned model is given by a sequence of Gaussian transitions $p_t(\theta_{t-1}|\theta_t, x) = \mathcal{N}(\theta_{t-1}|\mu_\psi(\theta_t, t, x), \beta_t)$ trained to invert a sequence of noising steps given by $q_t(\theta_t|\theta_{t-1}) = \mathcal{N}(\theta_t|\sqrt{1-\beta_t}\,\theta_{t-1}, \beta_t I)$. The connection between diffusion models and score-based methods comes from the fact that the optimal means and scores are linearly related [18]

$$\mu_\psi(\theta, t, x) = \frac{1}{\sqrt{\alpha_t}}\theta + \frac{1-\alpha_t}{\sqrt{\alpha_t}} s_\psi(\theta, t, x). \tag{11}$$

**Approximately composing $n$ diffusions** To simplify notation, we use a superscript $j$ to indicate distributions that are conditioned on $x_j$ (e.g. $p_t^j(\theta_t) = p_t(\theta_t|x_j)$). Assume we have transition kernels $p_t^j(\theta_{t-1}|\theta_t)$ that exactly reverse the forward kernels $q(\theta_t|\theta_{t-1})$ [A1], meaning that $p_{t-1}^j(\theta_{t-1}) = \int d\theta_t\, p_t^j(\theta_t)\, p_t^j(\theta_{t-1}|\theta_t)$, or equivalently $p_t^j(\theta_t)\, p_t^j(\theta_{t-1}|\theta_t) = p_{t-1}^j(\theta_{t-1})\, q_t(\theta_t|\theta_{t-1})$. Our goal is to find a transition kernel $\tilde{p}_t(\theta_{t-1}|\theta_t)$ that satisfies

$$\tilde{p}_{t-1}(\theta_{t-1}) = \int d\theta_t\, \tilde{p}_t(\theta_t)\, \tilde{p}_t(\theta_{t-1}|\theta_t), \tag{12}$$

where $\tilde{p}_t(\theta_t) = \frac{1}{Z_t}\prod_j^n p_t^j(\theta_t)$.[3] It is straightforward to verify that the condition from Eq. (12) can be re-written as

$$p_{t-1}^1(\theta_{t-1}) = \int d\theta_t\, p_t^1(\theta_t) \frac{p_t^2(\theta_t)}{p_{t-1}^2(\theta_{t-1})} \cdots \frac{p_t^n(\theta_t)}{p_{t-1}^n(\theta_{t-1})} \frac{Z_{t-1}}{Z_t} \tilde{p}_t(\theta_{t-1}|\theta_t) \tag{13}$$

$$= \int d\theta_t\, p_t^1(\theta_t) \frac{q_t(\theta_t|\theta_{t-1})}{p_t^2(\theta_{t-1}|\theta_t)} \cdots \frac{q_t(\theta_t|\theta_{t-1})}{p_t^n(\theta_{t-1}|\theta_t)} \frac{Z_{t-1}}{Z_t} \tilde{p}_t(\theta_{t-1}|\theta_t) \qquad \text{[A1]}. \tag{14}$$

[3] This is closely related to our formulation in Section 2, since our definition for the bridging densities involves the product $\prod_j^n p_t(\theta|x_j)$.

Then, one way to satisfy Eq. (14) is by setting $\tilde{p}_t(\theta_{t-1}|\theta_t)$ so that the term in blue above is equal to $p_t^1(\theta_{t-1}|\theta_t)$. That is,

$$\tilde{p}_t(\theta_{t-1}|\theta_t) = p_t^1(\theta_{t-1}|\theta_t)\frac{Z_t}{Z_{t-1}}\frac{p_t^2(\theta_{t-1}|\theta_t)}{q_t(\theta_t|\theta_{t-1})}\cdots\frac{p_t^n(\theta_{t-1}|\theta_t)}{q_t(\theta_t|\theta_{t-1})}. \tag{15}$$

However, the resulting $\tilde{p}_t(\theta_{t-1}|\theta_t)$ may not be a normalized distribution [30]. Following Sohl-Dickstein et al. [30], we propose to use the corresponding normalized distribution defined as $\tilde{p}_t^N(\theta_{t-1}|\theta_t) \propto \tilde{p}_t(\theta_{t-1}|\theta_t)$ [A2]. Given that Eq. (15) corresponds to the product of Gaussian densities, the resulting normalized transition is also Gaussian, with mean and variance given by

$$\mu_t = \frac{\sum_j \mu_{jt} - (n-1)\sqrt{\alpha_t}\theta}{n - \alpha_t(n-1)} \quad \text{and} \quad \sigma_t^2 = \frac{\beta_t}{n - \alpha_t(n-1)}, \tag{16}$$

where each $\mu_{jt}$ is obtained using Eq. (11).

**Prior correction term** The formulation above ignores the fact that the bridging densities defined in Eq. (3) involve the prior $p(\theta)$. We use the method proposed by Sohl-Dickstein et al. [30] to correct for this, which involves adding the term $\frac{\sigma_t^2(1-n)(T-t)}{T}\nabla_\theta \log p(\theta)$ to the mean $\mu_t$ from Eq. (16). (The derivation for this is similar to the one above, and also requires setting the resulting transition kernel to the normalized version of an unnormalized distribution [30].)

As mentioned previously, this derivation uses two assumptions/approximations. [A1] assumes that the learned score function/reverse diffusion approximately reverses the noising process, which is reasonable if the forward kernels $q_t$ add small amounts of noise per step (equivalently, if the noise levels $\gamma_1, \ldots, \gamma_T$ increase slowly). [A2] assumes that the normalized version of $\tilde{p}_t(\theta_{t-1}|\theta_t)$, given by $\tilde{p}_t^N(\theta_{t-1}|\theta_t)$, approximately satisfies Eq. (14).