Generative Modeling: A Review

Maria Nareklishvili*
Graduate School of Business
Stanford University

Nick Polson†
Booth School of Business
University of Chicago

Vadim Sokolov‡
Department of Industrial Engineering
George Mason University

First Draft: September 10, 2024
This Draft: June 26, 2026

Abstract

We organize the generative-modeling literature around three classes of generators, corresponding to three distinct inferential tasks: estimating counterfactual outcome distributions in causal inference, recovering posteriors from simulated parameter–outcome pairs, and forming predictive outcome distributions. The unifying representation relies on the noise outsourcing theorem of Kallenberg, which expresses a conditional distribution as a deterministic function of its inputs and an independent noise variable. Within this organization we develop generative Bayesian computation, a method in the parameter–outcome class: a quantile neural network, trained on simulated pairs under the pinball loss, that targets the posterior of the parameter directly, without invertible architectures or density evaluation, and that serves equally as a predictive generator once the roles of parameter and outcome are exchanged. We illustrate the framework on an agent-based Ebola transmission application, where generative Bayesian computation recovers accurate posteriors at substantially lower cost than rejection-based simulation inference, while avoiding the density-evaluation and invertibility constraints of competing generators.

Key words: generative Bayesian computation, simulation-based inference, quantile neural networks, causal inference, approximate Bayesian computation

*Maria Nareklishvili is a Postdoctoral Fellow at Stanford University, Graduate School of Business, email: marnar@stanford.edu

†Corresponding author; Nick Polson is Professor of Econometrics and Statistics at Chicago Booth, email: ngp@chicagobooth.edu

‡Vadim Sokolov is an Assistant Professor in Operations Research at George Mason University, email: vsokolov@gmu.edu

1 Introduction

Generative models provide a common language for several apparently distinct objectives in statistical inference: estimating counterfactual outcome distributions in causal inference, computing posterior distributions in Bayesian inference, and quantifying uncertainty in prediction. We organize this review around three generator classes that correspond to these three tasks. (1) Counterfactual outcome generators: recover the outcome map $Y = G(c, Z)$ with $Z \sim \text{Unif}(0, 1)$ and conditioning input c; potential outcomes in causal inference represent a special case $Y(a) = G(\tilde{S}(X), a, Z)$ with treatment state $a \in \{0, 1\}$, sufficient dimension reduction $\tilde{S}(X)$, from which causal estimands $\Delta = \mathcal{T}(G)$ are recovered. (2) Parameter–outcome generators: recover the inverse posterior map $(\theta \mid Y = y) \overset{d}{=} G(y, Z)$ from simulated parameter–outcome pairs, as an alternative to Markov chain Monte Carlo (Gelfand, 2000; Smith, 2007) or rejection-based approximate Bayesian computation (Beaumont et al., 2002; Sisson et al., 2018; Marin et al., 2012). (3) Predictive generators: recover the conditional outcome map $Y = G(X, Z)$ for prediction including its predictive uncertainty.

The three generator classes share a common representation theorem, the noise outsourcing theorem of Kallenberg (1997), which identifies each conditional distribution with a deterministic function of inputs and reference noise. The classes differ in what is targeted, in the data used to estimate the generator, and in the identification assumptions required: parameter generation requires a specified prior distribution and forward simulator, with consistency governed by the usual Bayesian posterior-concentration conditions; the counterfactual outcome generation additionally requires the Neyman-Rubin potential outcomes framework and the classical causal assumptions; and predictive inference requires only that training and target data share the same conditional distribution.

Alongside the review, we present generative Bayesian computation (GBC), which casts inference as a quantile regression estimated on simulated data. The key idea is to simulate the respective data set from the joint distribution: parameter–outcome pairs for parameter generation, input–output pairs for predictive inference, and conditional outcome triples for counterfactual outcome generation. A deep neural network is estimated to the relevant pair or triple type. Once fitted, the function estimator produces draws or quantiles at new inputs by direct evaluation, without further simulation or likelihood computation.

Formally, for parameter generation, we draw the training pairs $\{(\theta_i, Y_i)\}_{i=1}^N$ by forward simulation: one draws $\theta_i \sim p(\theta)$ from the prior distribution and then simulates $Y_i \sim p(Y \mid \theta_i)$ from the forward model. The goal is to learn a deterministic function G

such that

$$\theta = G(Y, Z), \quad Z \sim p(Z),$$

where Z is an auxiliary random variable drawn from a fixed reference distribution, for example uniform or standard normal, independent of the data. The function G transforms this reference noise into posterior draws conditional on Y. When $Z \sim \text{Unif}(0,1)$ and θ is scalar, this reduces to learning the inverse conditional quantile function $\theta = F^{-1}_{\theta|Y}(Z)$. The existence of such a representation is guaranteed by the noise outsourcing theorem (Kallenberg, 1997), discussed in Section 3. Applying the same theorem to the pair (c, Y) yields the outcome map $Y = G(c, Z)$ for outcome generation; the conditioning input c may be empty (unconditional generation), include a covariate vector (conditional generation), or include a treatment indicator A in which case the construction specializes to the counterfactual outcome map under standard assumptions in causal inference. Applying the theorem to the input–output pair (X, Y) yields the conditional outcome map $Y = G(X, Z)$ for predictive inference.

The number of simulated pairs N is the central control parameter of the framework, the analog of the reference-table size in approximate Bayesian computation, and is chosen by the researcher rather than fixed by the observed data; when simulation is computationally cheap, values of N in the range 10^4–10^6 are feasible; when simulation is costly, N is a design choice constrained by the computational budget. Larger N improves approximation accuracy at the cost of simulation effort, and the statistical properties of the resulting estimators are governed by N via approximation theory rather than by classical fixed-sample asymptotics. This simulation-based training strategy is applicable whenever a forward model is available, even when the likelihood function cannot be evaluated in closed form.

A natural objective for training is to estimate the conditional expectation $\hat{\theta}(Y_i) = \mathbb{E}[\theta \mid Y_i]$, which minimizes mean squared error and can be viewed as a nonparametric regression problem:

$$\theta_i = f(Y_i) + \varepsilon_i,$$

where ε_i is a mean-zero error term. This formulation targets the posterior mean $\mathbb{E}[\theta \mid Y_i]$; recovering the full posterior distribution requires the quantile-network extension below. Deep neural networks have emerged as effective approximators for f in high-dimensional and nonparametric regimes (LeCun et al., 2015; Jiang et al., 2017; Polson and Ročková, 2018; Montanelli and Yang, 2020; Schmidt-Hieber, 2020). To recover the full conditional distribution $p(\theta \mid Y)$ rather than just the posterior mean, one extends this framework by introducing an auxiliary random input $Z \sim \text{Unif}(0,1)$ and learning a function

$G(S(Y), \chi(Z))$, where χ is a learnable embedding of the reference noise (for example, the cosine quantile embedding of Section 5); this maps the data-dependent summary $S(Y)$ together with the embedded latent input into parameter space. Because the training sample size N is chosen by the researcher and can be made arbitrarily large, the statistical properties of the resulting estimators (including generalization error, convergence rates, and interpolation behavior) are governed by approximation theory rather than by fixed-sample asymptotics. Convergence rates for quantile neural networks are established by Shen et al. (2021) and Padilla et al. (2022), building on the deep nonparametric regression rates of Schmidt-Hieber (2020); the interpolation and double-descent literature (Belkin et al., 2019; Bach, 2024) provides additional asymptotic context. This framework is related to bootstrap methods (Efron, 1982, 1992; Efron and Tibshirani, 1994; DiCiccio and Efron, 1996), which approximate sampling distributions by repeatedly re-estimating the model on resampled data. Generative modeling differs in that it jointly simulates parameter–outcome pairs and fits the model only once, substantially reducing computation.

Generative Bayesian computation connects to several established lines of work: latent-variable models and auto-encoders (Albert et al., 2022; Akesson et al., 2021), simulation-based or implicit models (Diggle and Gratton, 1984; Baker et al., 2022; Schultz et al., 2022), and indirect inference, which constructs estimators by aligning features of observed and simulated data (Pastorello et al., 2003; Stroud et al., 2003; Drovandi et al., 2011, 2015). Each tradition developed with a distinct original purpose (data compression, density estimation, or parameter calibration), but all share the common structure of learning mappings between data and latent representations. We position these alternatives by asking which generator class they naturally serve and what changes when the same architecture is used in a different class.

The contributions of this paper are: (1) a three-class organization of generative inference into counterfactual outcome distributions, simulated parameter–outcome generators, and predictive generators, which we use as the spine of the methodological review (Section 2); (2) a common representation theorem, the noise outsourcing theorem of Kallenberg (1997), which expresses each class as a deterministic function of inputs and independent reference noise (the outcome map $Y = G(c, Z)$, the inverse posterior map $\theta = G(Y, Z)$, and the conditional outcome map $Y = G(X, Z)$), and identifies the choices of training pair, conditioning structure, and loss function that distinguish them (Sections 3.1 and 3.4); (3) a comparison of generative architectures by generator class, emphasizing training objects, outputs, and limitations (Section 4, with the cross-class synthesis in Section C and Tables 2, 3, and 4); (4) a worked instance for the parameter-outcome class, the generative Bayesian computation framework, which is a quantile neural net-

work trained by the pinball loss that requires no invertible architecture and no density evaluation, and that also serves as a predictive generator when the parameter and outcome roles are exchanged (Section 5, with the GBC-versus-GAN structural comparison in Table 6); and (5) three applications aligned with the three classes: Ebola inverse inference for the parameter-generation class (Section 6.1), Ebola predictive emulation for the predictive class (Section 6.2), and a synthetic program-evaluation example (no numerical results claimed) for counterfactual outcome distributions (Appendix F).

The remainder of the paper is organized as follows. Section 2 defines the three generator classes and states the common representation theorem. Section 3 develops the theoretical framework, establishing the noise outsourcing theorem for parameter generation and stating the analogous identification results for outcome generation and predictive inference. Section 4 reviews the generative architectures. Section 5 develops generative Bayesian computation. Section 6 details the application. Section 7 concludes with a decision guide and open challenges.

2 Three Generator Classes

We distinguish three classes of generators that share a common representation, a deterministic function of inputs and reference noise, but differ in their conditioning variables and target applications. The noise outsourcing theorem of Kallenberg (1997) provides the common foundation for all three.

The first class targets a new outcome variable given a (possibly empty) conditioning input c. The outcome map

$$Y = G(c, Z), \qquad Z \sim \text{Unif}(0, 1),$$

subsumes three common use cases: unconditional sample generation ($c = \emptyset$, the canonical task of generative adversarial networks and diffusion models for synthetic data), conditional generation (c representing a covariate or class label), and counterfactual generation (c includes a treatment indicator). Applied to the pair (c, Y), the noise outsourcing theorem identifies G pointwise as the generalized inverse of the conditional outcome distribution, $G(c, q) = F^{-1}_{Y|c}(q)$ for $q \in (0, 1)$ (Parzen, 2004).

When c includes a binary treatment state $a \in \{0, 1\}$ alongside a sufficient covariate reduction $\tilde{S}(X)$, the outcome map reduces to the *counterfactual generator*

$$G : (0, 1) \times \mathcal{S} \times \{0, 1\} \to \mathcal{Y}, \qquad q \mapsto G(q, \tilde{s}, a),$$

with $G(q, \tilde{s}, a) = F^{-1}_{Y|\tilde{S}(X)=\tilde{s},\, A=a}(q)$, non-decreasing and left-continuous in q. Under the standard causal assumptions (consistency $Y = Y(A)$, no interference, unconfoundedness $Y(a) \perp A \mid \tilde{S}(X)$, and positivity $0 < P(A = a \mid \tilde{S}(X) = \tilde{s}) < 1$ for $a \in \{0,1\}$), one obtains the identification equality $F_{Y(a)|\tilde{S}(X)=\tilde{s}} = F_{Y|\tilde{S}(X)=\tilde{s},\, A=a}$ for each a, so that $G(q, \tilde{s}, a) = F^{-1}_{Y(a)|\tilde{S}(X)=\tilde{s}}(q)$. The noise outsourcing theorem then recovers the potential outcomes $Y(a) = G(U, \tilde{S}(X), a)$, and any scalar causal estimand is a functional $\Delta = \mathcal{T}(G)$ of the generator. Conditional and average treatment effects are

$$\Delta(\tilde{s}) = \int_0^1 \big[G(q, \tilde{s}, 1) - G(q, \tilde{s}, 0)\big]\, dq, \qquad \Delta = \int \Delta(\tilde{s})\, dF_{\tilde{S}(X)}(\tilde{s}).$$

The unconditional and non-causal conditional distributions are recovered by setting $c = \emptyset$ or by taking c as a generic covariate vector without necessarily invoking commonly imposed identification assumptions in causal inference.

The second class targets the *Bayesian posterior* $p(\theta \mid Y)$. Applying the noise outsourcing theorem of Kallenberg (1997) to the pair (θ, Y) yields a measurable G such that

$$(\theta \mid Y = y) \overset{d}{=} G(y, Z), \qquad Z \sim \text{Unif}(0,1).$$

When θ is scalar, $G(y, \tau) = F^{-1}_{\theta|y}(\tau)$ is the inverse conditional cumulative distribution function. In the multivariate case, F^{-1} is not unique without an ordering convention; we adopt the autoregressive Rosenblatt-style factorization of Section 5, while alternative vector-quantile constructions based on the Brenier optimal-transport map provide a coordinate-free generalization (Carlier et al., 2016). Section 3 develops this representation and its sufficient-statistic and quantile-based forms; Section 5 presents generative Bayesian computation, a parameter-generation specialization built on the pinball loss.

The third class targets the *predictive distribution* $p(Y \mid X)$. Applying the noise outsourcing theorem to the pair (X, Y) yields

$$Y \mid X = x \overset{d}{=} G(x, Z), \qquad Z \sim \text{Unif}(0,1),$$

where G plays the role of a forward emulator. Quantile networks trained with the pinball loss are well suited to this predictive class; Section 6 demonstrates this construction in the Ebola application by predicting 56-week infection trajectories from model parameters with full predictive uncertainty.

Notation. For parameter generation, θ denotes the Bayesian parameter, Y the data, and Z auxiliary noise; $\tau \in (0,1)$ denotes a quantile level. For predictive inference, X

denotes covariates, Y the outcome, and Z auxiliary noise. For outcome generation, c denotes a possibly empty conditioning input and Z auxiliary noise; the counterfactual outcome generation adds a binary treatment state $a \in \{0, 1\}$, a treatment indicator A, and a sufficient covariate reduction $\tilde{S}(X)$, with $U \sim \text{Unif}(0, 1)$ for the latent rank index and $q \in (0, 1)$ for a fixed quantile level. The symbols q and τ play the same mathematical role (a fixed level in $(0, 1)$); we use τ in the parameter-generation sections following the convention of the generative Bayesian computation and q in the sections describing counterfactual outcome generation following the convention in causal inference. We reserve θ for Bayesian model parameters; causal estimands are denoted by Δ.

3 Generative Framework

This section formalizes the generative approach to posterior inference. The central idea is to represent the posterior distribution $p(\theta \mid Y)$ through a deterministic function of the observed data and an independent source of randomness, rather than through density evaluation or iterative sampling. Throughout, we use N for the number of simulated training pairs, n for the number of observations in a statistical model, and τ for quantile levels. The noise outsourcing theorem of Section 3.1 gives the parameter-generation mechanism; the analogous identification results for outcome generation and predictive inference are stated in Section 3.4.

Consider a Bayesian model defined by a prior $p(\theta)$ and a forward model $p(Y \mid \theta)$. We introduce an auxiliary latent variable $Z \sim p(Z)$ drawn from a known reference distribution (for example, $Z \sim \text{Unif}(0, 1)$) independently of both θ and Y. The goal is to find a measurable function G such that

$$\theta = G(Y, Z), \quad Z \sim p(Z), \tag{1}$$

and the conditional distribution of $G(Y, Z)$ given $Y = y$ matches the posterior $p(\theta \mid Y = y)$ for all y. The existence of such a function is guaranteed under mild conditions by the noise outsourcing theorem, described below. Once G is learned from simulated training pairs $\{(\theta_i, Y_i)\}_{i=1}^{N}$, approximate posterior samples for any new observation y are obtained by drawing $Z_1, \ldots, Z_M \sim p(Z)$ and evaluating $G(y, Z_j)$ for $j = 1, \ldots, M$, where M is the desired number of posterior draws, without further simulation or likelihood evaluation.

3.1 Noise Outsourcing Theorem

Let (Y, θ) be random variables defined on a Borel space $(\mathcal{Y} \times \Theta)$. The noise outsourcing theorem (Kallenberg, 1997, Theorem 5.10) guarantees the existence of a measurable function $G : \mathcal{Y} \times [0,1] \to \Theta$ and a random variable $Z \sim \text{Unif}(0,1)$, independent of Y, such that

$$(Y, \theta) \overset{a.s.}{=} (Y, G(Y, Z)).$$

Equivalently, for any $y \in \mathcal{Y}$, the conditional distribution of $G(y, Z)$ given $Y = y$ equals the posterior $p(\theta \mid Y = y)$:

$$(\theta \mid Y = y) \overset{d}{=} G(y, Z), \quad Z \sim \text{Unif}(0,1).$$

This result, also known as the functional representation lemma (Kallenberg, 1997), provides a unified view of conditional distribution estimation and nonparametric regression: finding the posterior $p(\theta \mid Y)$ is equivalent to finding a deterministic function G of the data and independent noise. The function G is measurable but not necessarily differentiable; when θ is scalar, it coincides with the inverse conditional cumulative distribution function, $G(y, \tau) = F^{-1}_{\theta|y}(\tau)$. The theorem requires that $(\mathcal{Y} \times \Theta)$ be a standard Borel space, which holds for Euclidean spaces and most spaces encountered in statistical applications. It does not require continuity of the posterior or existence of a density; however, when the posterior is discrete or singular, the resulting function G may be discontinuous in τ, which can complicate approximation by smooth neural networks.

The same theorem is invoked across all three generator classes, so what it actually delivers must be stated precisely, both to prevent reading more into the "single foundation" than is there and to fix the perimeter for the identification and approximation arguments that follow. The theorem is a statement of representation and existence: it guarantees that a generator G exists and characterizes it pointwise. It supplies no estimation procedure: it does not specify how to learn G from a finite simulated data set. It supplies no rates: approximation quality is a distinct question, governed by the smoothness of G and the network theory of Section 5. Additionally, it does not guarantee identification: in the counterfactual class it is the causal assumptions (consistency, unconfoundedness, positivity, the stable unit treatment value assumption) that license interpreting G as a potential-outcome map, not the theorem. The unification across the three classes is therefore representational: one object, applied to three pairs. It is not an inferential unification, and the identification and approximation content of each class must be argued on its own terms.

3.2 Sufficient Statistics and Dimensionality Reduction

When the observed data Y is high-dimensional, the generator G can be estimated more efficiently by conditioning on a low-dimensional summary $S(Y)$ rather than on Y directly. A statistic $S(Y)$ is sufficient in the Bayesian sense (Kolmogorov, 1942) if, for any prior $p(\theta)$, the posterior satisfies $p(\theta \in B \mid Y) = p(\theta \in B \mid S(Y))$ for all measurable $B \subseteq \Theta$. When such a sufficient statistic exists and satisfies $Y \perp \theta \mid S(Y)$, the noise outsourcing theorem simplifies to

$$\theta \mid Y \overset{d}{=} G(S(Y), Z).$$

In parametric models, sufficient statistics are often available in closed form. For forward models with intractable likelihoods, sufficient statistics must be approximated. Approaches include deep neural networks (Jiang et al., 2017), autoencoders, partial least squares (Polson et al., 2021; Brillinger, 2012; Bhadra et al., 2021), and kernel-based embeddings (Park et al., 2016). The choice of summary statistic is critical: Beaumont et al. (2002) and Nunes and Balding (2010) discuss optimal selection in exponential families, while Jiang et al. (2018) and Bernton et al. (2019) propose smoothing-based extensions for approximate Bayesian computation. Jiang et al. (2017) show that the posterior mean $\mathbb{E}[\theta \mid Y]$ is an optimal summary under quadratic loss and provide asymptotic guarantees for deep network estimators of $S(Y)$.

3.3 Quantile-Based Posterior Estimation

Given training pairs $\{(\theta_i, Y_i)\}_{i=1}^N$ simulated from the joint distribution (the analog of an approximate Bayesian computation reference table, with N chosen by the researcher), the posterior can be estimated through its quantile function. We adopt the quantile-function viewpoint of Parzen (2004), who shows that quantile methods are direct alternatives to density estimation and provides three properties we use throughout: (i) the conditional quantile $Q_{\theta|Y}(\tau, y) = F^{-1}_{\theta|y}(\tau)$ is non-decreasing and left-continuous in τ; (ii) it satisfies the identity $\theta \overset{d}{=} Q_{\theta|Y}(Z, Y)$ when $Z \sim \text{Unif}(0,1)$, providing a sampler that requires only one uniform draw and one network evaluation per posterior sample; and (iii) quantile composition, $Q_{g(\theta)|Y}(\tau, y) = g(Q_{\theta|Y}(\tau, y))$ for non-decreasing left-continuous g, makes quantile-based posteriors invariant to monotone reparameterizations and is used in Appendix B to derive a quantile-form Bayes rule. A deep neural network $\hat{Q}(\tau, y; w)$ is esti-

mated to approximate this quantile function by minimizing the pinball (quantile) loss:

$$\mathcal{L}(w) = \frac{1}{NK} \sum_{i=1}^{N} \sum_{k=1}^{K} \rho_{\tau_k}\big(\theta_i - \hat{Q}(\tau_k, Y_i; w)\big), \quad \rho_\tau(u) = u(\tau - \mathbb{1}_{u<0}),$$

where ρ_τ is the asymmetric check function (Koenker and Bassett Jr, 1978). By training over a grid of quantile levels or by treating τ as an additional input to the network (Dabney et al., 2018; Ostrovski et al., 2018), one can approximate the full posterior distribution. Unlike approximate Bayesian computation, which relies on rejection sampling and becomes inefficient in high-dimensional settings, this approach evaluates the trained network directly at the observed data. The quality of the approximation depends on the smoothness of the posterior map and the generalization ability of the network; see Padilla et al. (2022) and Moon et al. (2021) for convergence rates of quantile neural networks. Appendix A characterizes Bayes risk bounds for such estimators.

3.4 Mechanisms for Outcome Generation and Predictive Inference

The noise outsourcing theorem of Kallenberg (1997), stated in Section 3.1, extends to outcome generation and predictive inference. Applied to the pair (c, Y), where c is a (possibly empty) conditioning input, the theorem yields a Borel-measurable G such that

$$Y \mid c \overset{d}{=} G(c, Z), \qquad Z \sim \text{Unif}(0, 1),$$

identified pointwise via the generalized inverse $G(c, q) = F^{-1}_{Y|c}(q)$, following Parzen (2004). Three distinct cases recover the canonical applications: (i) unconditional sample generation when $c = \emptyset$, where $Y \overset{d}{=} G(Z)$ produces synthetic samples from the marginal data distribution; (ii) conditional generation when c is a covariate or class label, providing $Y \mid c$ for synthetic-data problems; and (iii) counterfactual generation when c includes a treatment indicator $A \in \{0, 1\}$ alongside a sufficient covariate reduction $\tilde{S}(X)$, yielding the counterfactual outcome map $Y(a) = G(U, \tilde{S}(X), a)$ under commonly invoked assumptions in causal inference (consistency, unconfoundedness given $\tilde{S}(X)$, positivity, and the stable unit treatment value assumption). Under these assumptions, any scalar causal estimand is identified as a functional $\Delta = \mathcal{T}(G)$ of the generator, including the conditional and average treatment effects.

The conditional outcome map in predictive inference admits the same noise-outsourcing

representation with (X, Y) in place of (c, Y):

$$Y \mid X = x \overset{d}{=} G(x, Z), \qquad Z \sim \text{Unif}(0, 1),$$

where $G(x, \tau) = F^{-1}_{Y|X=x}(\tau)$ is the conditional quantile function. Quantile neural networks trained with the pinball loss (Section 3.3) are the natural estimators for predictive inference: the same estimation procedure as in parameter generation, with the roles of θ and Y exchanged. One conceptual caution: in parameter generation the training pairs are drawn from the product of prior and simulator and Z encodes *epistemic* posterior uncertainty, whereas in prediction the pairs come from observed (or simulated) data and Z encodes *aleatoric* outcome variability. The symmetry is architectural; the source of the training distribution, and the interpretation of Z, differ.

4 Generator Classes and Associated Methods

The three generator classes share a common representation, noise outsourcing theorem of Kallenberg (1997) applied to different variable pairs, but differ in what they condition on, which identification assumptions they require, and which statistical question they answer. The architecture of the method itself does not determine class membership; the training object and conditioning structure do, hence, any architecture that can represent a conditional distribution can in principle serve any of the three classes once its loss, conditioning variables, and output interpretation are changed accordingly. For example, the unconditional adversarial or diffusion generator, built to match a data distribution, is therefore primarily an outcome generator; conditioning it on covariates redirects it to prediction, and conditioning it on a treatment indicator under causal assumptions redirects it to counterfactual outcome generation. Approximate Bayesian computation and fiducial inference invert a forward data-generating equation and are for that reason tied to the parameter–outcome class, with no natural counterfactual or predictive analogue. A single architecture can thus serve several classes: a method may estimate one target without retraining (its primary class) and reach another only after re-conditioning or a change of loss (a secondary role). The subsections below provide a detailed formulation of each architecture under the class it primarily serves. Appendix C compares the methods in greater detail, with tables displaying training object, output, and limitations of each class; the primary/secondary class membership of the generative architectures; and their computational properties.

4.1 Generators for Counterfactual Outcome Distributions

The first generator class targets both observed or counterfactual outcome distributions. The generator of potential outcomes is defined as

$$Y(a) = G(U, \tilde{S}(X), a), \qquad U \sim \text{Unif}(0,1), \quad a \in \{0,1\},$$

where $\tilde{S}(X)$ is a sufficient covariate reduction and a indexes the treatment state. In non-causal applications the same class reduces to conditional or unconditional outcome generation, $Y = G(c, Z)$, where c is empty, a class label, or a vector of covariates. The statistical target is the conditional distribution of outcomes, not a posterior over model parameters.

Generative adversarial networks (Goodfellow et al., 2014, 2020) and diffusion models are natural members of this class. An adversarial generator learns to transform noise and conditioning variables into synthetic outcomes by matching the generated and observed distributions through a discriminator. This is useful for rich outcome spaces, but the adversarial objective does not by itself guarantee calibrated counterfactual intervals or valid causal identification. Diffusion models, in their discrete-time (Sohl-Dickstein et al., 2015; Ho et al., 2020) and continuous-time score-based (Song et al., 2021) formulations, primarily generate outcomes: they learn a reverse denoising process that maps Gaussian noise into draws from the outcome distribution. Their advantage is stable generation of multimodal distributions; their cost is many denoising steps per draw, although consistency-model variants reduce this cost (Song et al., 2023). In the causal setting, GANITE (Yoon et al., 2018) adapts the adversarial construction to individualized treatment effects, and double machine learning (Chernozhukov et al., 2018) provides an orthogonalized alternative for average effects.

Conditional variational autoencoders and conditional normalizing flows can also be used for outcome generation. A conditional variational autoencoder represents unobserved heterogeneity with a latent variable and decodes it into outcomes given covariates or treatment state. This gives a flexible generator, but the Gaussian latent approximation may understate tail behavior or multimodality. A conditional flow provides exact density evaluation when the map is invertible and dimensions match, but the bijection requirement can be restrictive in causal problems where the outcome dimension, covariate dimension, and latent heterogeneity need not align.

Quantile generators provide a particularly direct formulation for counterfactual distributions. Estimating $G(q, \tilde{s}, a) = F^{-1}_{Y|\tilde{S}(X)=\tilde{s}, A=a}(q)$ by the pinball loss yields the full conditional quantile process. Average effects, conditional effects, and quantile treatment effects (Chernozhukov and Hansen, 2005) are then functionals of the same fitted generator. This

approach separates the statistical identification assumptions from the numerical architecture: unconfoundedness and positivity identify the counterfactual distribution, while the generator estimates its conditional quantile map.

We now give the detailed formulation of the architectures whose primary home is this class: generative adversarial networks, diffusion models, and the decoder of the variational autoencoder. Each is presented in its own established notation, with the symbol θ noted explicitly whenever it denotes network parameters rather than the Bayesian parameter of interest.

Generative adversarial networks

The generative adversarial network of Goodfellow et al. (2020) learns an implicit outcome distribution by training two networks in competition; here θ_g and θ_d are network parameters, not the Bayesian parameter of interest. In its original form the generator $G_{\theta_g}(Z)$ maps noise $Z \sim p(Z)$ to a synthetic outcome and matches the marginal data distribution, so the canonical generative adversarial network is an *unconditional outcome generator*; conditioning the generator as $G_{\theta_g}(Z, X)$ redirects it to a predictive target when X is a covariate vector, and supplying a treatment indicator a as in $G_{\theta_g}(Z, X, a)$ redirects it to a counterfactual outcome prediction. A discriminator D_{θ_d} classifies samples as real or generated. The networks are trained by maximizing the value function

$$J(\theta_d, \theta_g) = \mathbb{E}_Y[\log D_{\theta_d}(Y)] + \mathbb{E}_Z[\log(1 - D_{\theta_d}(G_{\theta_g}(Z)))], \tag{2}$$

whose first term is estimated by an empirical average over observed training samples. Under a zero-sum formulation the parameters solve the minimax problem $\min_{\theta_g} \max_{\theta_d} J(\theta_d, \theta_g)$, so the generator learns to produce samples the discriminator cannot distinguish from real data. A GAN produces a synthetic outcome $\tilde{Y} = G_{\theta_g}(Z)$ (Figure 1), and its target can be written through the Jensen–Shannon divergence between the observed law p_{obs} and the generated law p_G,

$$\text{JS}(p_{\text{obs}}, p_G) = \log 2 + \tfrac{1}{2} \sup_{D:\mathcal{Y}\to[0,1]} \big(\mathbb{E}_{Y\sim p_{\text{obs}}}[\log D(Y)] + \mathbb{E}_{Y\sim p_G}[\log(1 - D(Y))]\big).$$

The discriminator thus plays the role that the ϵ-neighborhood plays in approximate Bayesian computation: it supplies the criterion for deciding whether the generated distribution is close to the observed one.

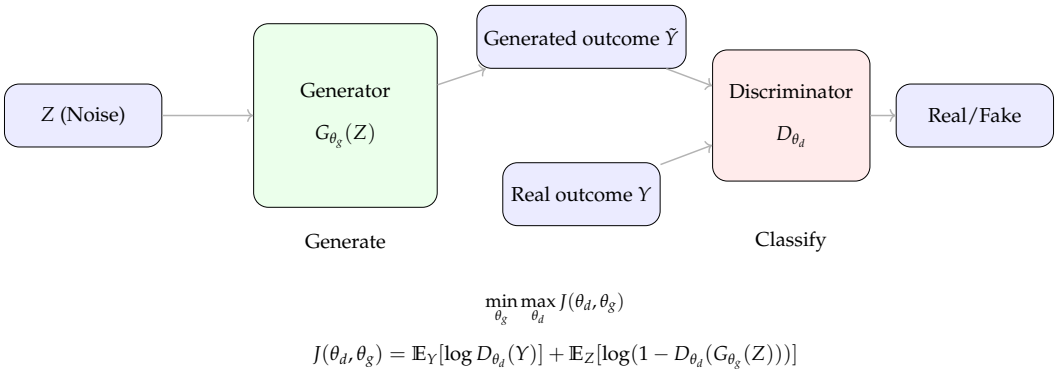

Figure 1: Generative adversarial network. The generator maps noise Z to a synthetic outcome $\tilde{Y}$; the discriminator scores outcomes as real (Y) or generated ($\tilde{Y}$), and the two networks are trained against the minimax objective.

Consider a study that records patient outcomes Y under treatments A together with characteristics X such as age and biomarkers. A conditional GAN (Mirza and Osindero, 2014) generates synthetic outcomes $\tilde{Y} = G_{\theta_g}(Z, X, a)$ by transforming noise $Z \sim \mathcal{N}(0, I)$ and patient features into plausible responses, while the discriminator evaluates whether an outcome is consistent with the patient profile and treatment assignment. Conditioned on a treatment indicator, the generator instantiates the counterfactual outcome map $G(U, X, a)$ of Section 2 (with the reference noise Z playing the role of the latent rank U), and treatment-effect functionals $\Delta = \mathcal{T}(G)$ such as the conditional average treatment effect follow from sampling different noise vectors for a fixed patient. Athey et al. (2022) provide a complementary application: GAN-generated profile images on a micro-lending platform isolate the causal effect of modifiable style features (smiling, framing) from fixed characteristics on funding outcomes.

At the minimax optimum $p_G = p_{\text{obs}}$, the generator reproduces the data distribution exactly. Unlike normalizing flows, which minimize a forward Kullback–Leibler divergence by maximum likelihood, and variational autoencoders, which optimize an evidence lower bound, the Jensen–Shannon objective requires no density evaluation and no parametric assumption on p_{obs}, so GANs apply to implicit models where only sampling is possible. The cost is training instability: the minimax problem can cycle rather than converge (Salimans et al., 2016), and the generator can suffer *mode collapse*, concentrating mass on a subset of the modes of p_{obs}, because the Jensen–Shannon penalty discourages mass outside the support of p_{obs} but does not penalize missing modes. This makes GANs less reliable than diffusion models for covering a multimodal outcome distribution, despite their faster sampling. A conditional GAN extends to predictive inference, but because it produces samples rather than a density, predictive intervals must be read off the gen-

erated sample and coverage assessed empirically; the discriminator yields no calibrated parameter posterior, so the GAN is not a natural parameter–outcome estimator.

Diffusion models

Diffusion models are transport-based generators built from a forward noising process, a learned reverse denoising process, and a sampling procedure (Ho et al., 2020; Song et al., 2021); this paragraph writes Y_t for the outcome sample at diffusion step t and ϕ for network parameters. The forward process corrupts a data point $Y_0 \sim q(Y_0)$ with Gaussian noise over T steps,

$$q(Y_{1:T} \mid Y_0) = \prod_{t=1}^{T} q(Y_t \mid Y_{t-1}), \qquad q(Y_t \mid Y_{t-1}) = \mathcal{N}(\sqrt{1-\beta_t}\, Y_{t-1},\, \beta_t I),$$

with variance schedule β_t, so that $q(Y_T)$ approaches an isotropic Gaussian for large T. The product $\prod_{t=1}^{T} q(Y_t \mid Y_{t-1})$ is the joint density of the entire noising trajectory $Y_1, \ldots, Y_T$ conditional on the noise-free (uncorrupted) data point Y_0, factorized into one-step transitions by the Markov property of the forward process. The reverse process learns a denoising model that inverts the corruption,

$$p_\phi(Y_{0:T}) = p(Y_T) \prod_{t=1}^{T} p_\phi(Y_{t-1} \mid Y_t), \qquad p(Y_T) = \mathcal{N}(0, I).$$

Training maximizes a variational bound on the data log-likelihood that reduces to denoising score matching: with closed-form marginal $q(Y_t \mid Y_0) = \mathcal{N}(\sqrt{\bar{\alpha}_t}\, Y_0,\, (1-\bar{\alpha}_t) I)$ and $\bar{\alpha}_t = \prod_{s \le t}(1-\beta_s)$, the network s_ϕ minimizes

$$\min_{\phi} \sum_{t=1}^{T} \lambda_t \, \mathbb{E}\big[\| s_\phi(Y_t, t) - \nabla_{Y_t} \log q(Y_t \mid Y_0) \|^2\big],$$

so $s_\phi(Y_t, t)$ approximates the score $\nabla_{Y_t} \log q_t(Y_t)$ of the noised data distribution, the same object used in score-based estimation (Hyvärinen, 2005). Song et al. (2021) show that the forward and reverse processes solve stochastic differential equations whose reverse direction depends on this score; sampling starts from $Y_T \sim \mathcal{N}(0, I)$ and integrates the reverse process to produce $\hat{Y}_0$. Because each step is a local correction rather than a global bijection, diffusion models handle well-separated modes better than normalizing flows, at the cost of hundreds of network evaluations per draw, which consistency-model variants reduce (Song et al., 2023). The multimodal energy landscape of protein-conformation

generation is a canonical setting where this mode coverage makes diffusion preferable to flows. The same construction targets a posterior when the diffusing variable is the parameter θ rather than the outcome: conditioning the score on outcome Y, Geffner et al. (2023) decompose $\nabla_\theta \log p_t(\theta \mid Y) = \nabla_\theta \log p_t(\theta) + \nabla_\theta \log p_t(Y \mid \theta)$ for compositional approximation and Sharrock et al. (2024) train on simulated pairs with sequential refinement, while classifier-free guidance (Ho and Salimans, 2022) conditions on covariates for prediction.

Variational autoencoders

A variational autoencoder (Kingma and Welling, 2014; Kingma et al., 2019) is a latent-variable generative model that learns both a low-dimensional representation of high-dimensional data and a mechanism for generating new samples. We present it in the paper's convention, where $Y \in \mathcal{Y}$ denotes an outcome, $Z \in \mathcal{Z}$ a latent variable, and (w, ϕ) the decoder and encoder network parameters, respectively (reserving θ for the Bayesian parameter of interest). The variational autoencoder mirrors the Bayesian setup with the latent $Z \leftrightarrow \theta$ playing the role of the parameter: the encoder approximates the posterior $p(\theta \mid Y)$ while the decoder corresponds to the forward model $p(Y \mid \theta)$; the difference is that the variational autoencoder assumes a fixed latent prior $p(Z) = \mathcal{N}(0, I)$ and optimizes a variational bound, whereas in Bayesian inference the prior $p(\theta)$ is part of the model.

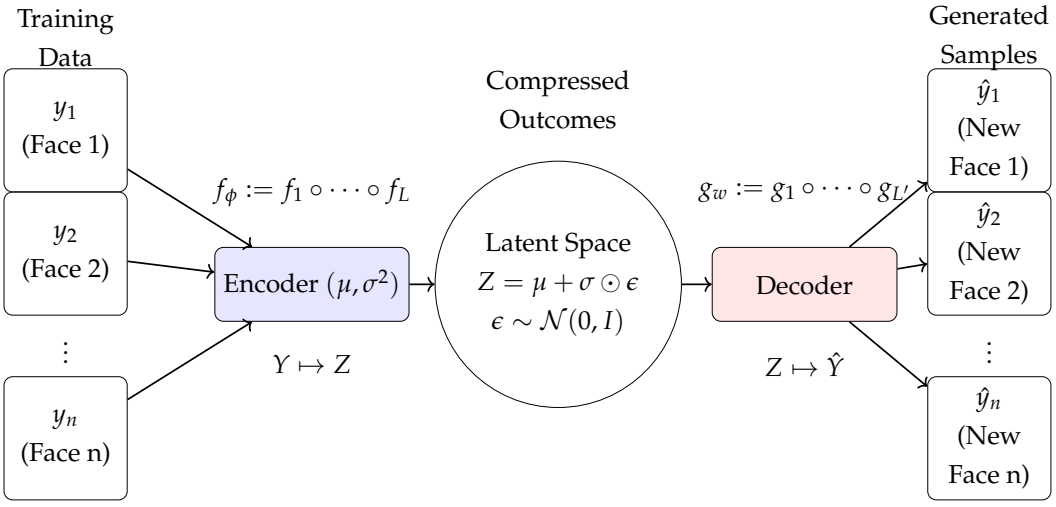

Figure 2: Variational autoencoder. The encoder (f_ϕ parameterizing q_ϕ) and decoder (g_w parameterizing p_w) are composite maps between outcome space (Y) and latent space (Z). The encoder compresses an enumerated training sample $y_1, \ldots, y_n$ into latent representations, from which the decoder generates new examples $\hat{y}_1, \ldots, \hat{y}_n$.

This generative model has two parts: a latent prior $Z \sim p(Z) = \mathcal{N}(0, I)$ representing mean-zero independent unobservables, and a conditional outcome model $Y \mid Z \sim p_w(Y \mid Z)$ in which w indexes a flexible function such as a neural network. The encoder targets the latent posterior $p_w(Z \mid Y) = p_w(Y \mid Z)\, p(Z)/p_w(Y)$, whose marginal likelihood $p_w(Y)$ is in general intractable. The variational autoencoder therefore introduces a Gaussian variational approximation $q_\phi(Z \mid Y) = \mathcal{N}(Z \mid \mu, \mathrm{diag}(\sigma^2))$ with $\mu = \mu_\phi(Y)$ and $\sigma = \sigma_\phi(Y)$ output by an encoder network, and uses the reparameterization $Z = \mu + \sigma \odot \epsilon$, $\epsilon \sim \mathcal{N}(0, I)$, so that sampling is a differentiable function of (μ, σ) and the decoder reconstruction $\hat{Y} = g_w(Z)$ can be optimized by gradient descent (see Figure 2). Training maximizes the evidence lower bound[1]

$$\mathcal{L}(w, \phi, y) = \mathbb{E}_{z \sim q_\phi(\cdot \mid y)}\big[\log p_w(y \mid z)\big] - \mathrm{KL}\big(q_\phi(z \mid y) \,\|\, p(z)\big), \tag{3}$$

with full objective $\mathbb{E}_{Y \sim p_{\mathrm{obs}}}[\mathcal{L}(w, \phi, Y)]$, estimated by its average over the training sample.

We note that the decoder $p_w(Y \mid Z)$ is the outcome generator: drawing $Z \sim p(Z)$ and decoding yields synthetic draws from the marginal data distribution, and a conditional variational autoencoder that appends a treatment indicator instantiates the counterfactual outcome map $G(U, X, a)$ when the Gaussian latent is an adequate model for the potential-outcome distribution. The encoder is instead an approximate posterior, placing the same architecture in the parameter–outcome class; because the variational family is Gaussian, multimodal or heavy-tailed posteriors are systematically underdispersed regardless of network capacity, a limitation not shared by the likelihood-free methods of that class. A conditional variational autoencoder that takes covariates as input produces a predictive distribution $p(Y \mid X)$ with calibrated intervals only when the Gaussian latent structure matches the predictive law, in contrast to quantile networks, which make no such assumption. Read against classical structure, the variational autoencoder is a nonlinear factor-analysis model: the Gaussian latent prior is a modeling convention rather

[1]For a sample $y \sim p_{\mathrm{obs}}$ and any variational density $q_\phi(z \mid y)$ that integrates to one, the marginal likelihood admits the identity

$$p_w(y) = \int \frac{p_w(y, z)}{q_\phi(z \mid y)}\, q_\phi(z \mid y)\, dz = \mathbb{E}_{z \sim q_\phi(\cdot \mid y)}\Big[\tfrac{p_w(y,z)}{q_\phi(z \mid y)}\Big].$$

Applying Jensen's inequality to the concave log,

$$\log p_w(y) = \log \mathbb{E}_{z \sim q_\phi(\cdot \mid y)}\Big[\tfrac{p_w(y,z)}{q_\phi(z \mid y)}\Big] \geq \mathbb{E}_{z \sim q_\phi(\cdot \mid y)}\Big[\log \tfrac{p_w(y,z)}{q_\phi(z \mid y)}\Big] =: \mathcal{L}(w, \phi, y).$$

Because $p_w(y, z) = p_w(y \mid z)\, p(z)$, taking expectations under $q_\phi(z \mid y)$ yields the bound above, where $\mathrm{KL}(\cdot \| \cdot)$ is the Kullback–Leibler divergence. Maximizing it tightens the gap to $\log p_w(y)$ and drives $q_\phi(z \mid y)$ toward the true latent posterior $p_w(z \mid y)$.

than the inferential prior, and the decoder is a nonlinear analogue of the factor-loading map.

4.2 Generators from Simulated Parameter–Outcome Pairs

The second generator class targets Bayesian inverse inference. The data object is a simulated reference table

$$\{(\theta_i, Y_i)\}_{i=1}^{N}, \qquad \theta_i \sim p(\theta), \quad Y_i \sim p(Y \mid \theta_i),$$

and the target is a generator for the posterior distribution,

$$(\theta \mid Y = y) \overset{d}{=} G(y, Z), \qquad Z \sim \mathrm{Unif}(0, 1).$$

This is the class most directly associated with simulation-based inference (Cranmer et al., 2020). The generator consumes an observed outcome or summary statistic and returns posterior draws, posterior quantiles, or a posterior density approximation.

The two classical members are approximate Bayesian computation and fiducial inference. Approximate Bayesian computation generates parameter–outcome pairs from the prior and forward simulator, then keeps parameters whose simulated outcomes are close to the observed outcome. It requires no likelihood evaluation and is broadly applicable, but its acceptance rate deteriorates quickly with the dimension of the summary statistic. Fiducial inference also inverts a forward data-generating equation, but it treats the resulting distribution over parameters as induced by the structural equation rather than by a prior. The connection to the present framework is structural and worth stating directly: the noise outsourcing representation $\theta = G(Y, Z)$ *is* an inverted data-generating equation, and modern generalized fiducial inference (Hannig et al., 2016) and the theory of confidence distributions (Xie and Singh, 2013) study exactly such inversions and the conditions under which the resulting distribution has calibrated frequentist coverage. In regular problems the fiducial distribution can coincide with the Bayesian posterior under Jeffreys' prior; outside such settings it requires additional conventions. A generator trained on (θ, Y) pairs and a generalized fiducial construction can therefore be read as two routes to the same object, and the fiducial literature is a natural source of coverage guarantees for the parameter–outcome class. Approximate Bayesian computation and fiducial inference are tied to this class because the simulated table $\{(\theta_i, Y_i)\}$ is their natural training object; normalizing flows are bidirectional and serve this class through the same change-of-variables framework used for outcome and predictive generation.

The modern simulation-based inference literature replaces rejection with a trained network. Neural posterior estimation fits a conditional density (typically a normalizing flow) directly to the posterior (Papamakarios and Murray, 2016; Greenberg et al., 2019); the BayesFlow framework (Radev et al., 2020) pairs an invertible flow with a summary network learned jointly from the simulator and is widely used for amortized Bayesian inference. These approaches yield exact density evaluation at the price of an invertible architecture and a tractable Jacobian. Neural likelihood estimation instead models $p(Y \mid \theta)$ (Papamakarios et al., 2019), and neural ratio estimation targets the likelihood-to-evidence ratio by classification (Hermans et al., 2020; Durkan et al., 2020); both then require a downstream sampler. Conditional diffusion models learn a posterior score and generate parameter draws through iterative denoising (Geffner et al., 2023; Sharrock et al., 2024), handling multimodality at substantial prediction-time cost. Software for this family is mature (Tejero-Cantero et al., 2020), and standardized benchmarks exist (Lueckmann et al., 2021).

Generative Bayesian computation belongs to this same parameter–outcome class but uses a quantile rather than density parameterization. For scalar θ, it estimates $F^{-1}_{\theta \mid Y=y}(\tau)$ directly with the pinball loss; for vector parameters, it uses an autoregressive conditional quantile factorization. It therefore requires neither likelihood evaluation nor an invertible map. The price is that it returns posterior draws and quantiles rather than a closed-form posterior density, so calibration checks such as coverage diagnostics remain essential (Section 6.1). The detailed formulation of the classical members and the bidirectional flow follows.

Approximate Bayesian computation

Approximate Bayesian computation conducts likelihood-free inference when data can be simulated from a structural model (Beaumont et al., 2002; Marin et al., 2012; Sisson et al., 2018). It simulates a reference table $\{(\theta_i, Y_i^\star)\}_{i=1}^N$ with $Y_i^\star \sim p(Y \mid \theta_i)$ and compares each simulated output to the data through a low-dimensional summary statistic $S(\cdot) \in \mathbb{R}^k$, $k \ll \dim(Y)$. A common posterior approximation is

$$p^\epsilon_{\mathrm{ABC}}(\theta \mid Y = y) = \frac{1}{m_\epsilon(y)} \int K_\epsilon(S(y^\star) - S(y))\, p(y^\star \mid \theta)\, p(\theta)\, dy^\star, \tag{4}$$

where K_ϵ is a kernel with bandwidth ϵ and $m_\epsilon(y)$ a normalizing constant. As $\epsilon \to 0$ with S sufficient, $p^\epsilon_{\mathrm{ABC}}(\theta \mid Y) \to p(\theta \mid Y)$; with a uniform kernel the approximation reduces to the conditional law of θ given $\|S(Y^\star) - S(Y)\| < \epsilon$, so that $p^\epsilon_{\mathrm{ABC}}(\theta \mid Y) \propto p(\theta)\, p(\|S(Y^\star) - S(Y)\| < \epsilon \mid \theta)$. Figure 3 summarizes the construction. The table size N is the central

control parameter: a larger table sustains a smaller bandwidth at a fixed acceptance rate, the same role N plays in the generative methods of Section 5. A central application is population genetics, where likelihood-based inference for demographic processes such as migration is intractable but genetic data can be simulated under competing scenarios and compared through summary statistics such as allele frequencies.

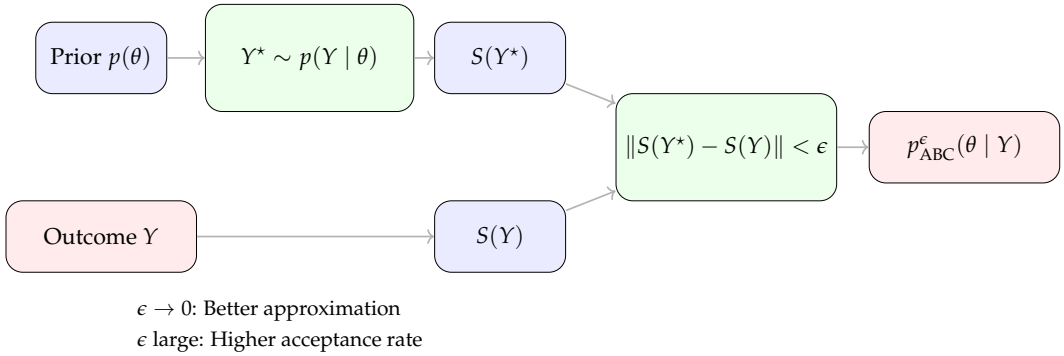

Figure 3: Approximate Bayesian computation. Parameters drawn from the prior are pushed through the simulator; a parameter is retained when its simulated summary statistic falls within ϵ of the observed summary statistic.

Jiang et al. (2017) prove that the posterior mean $\mathbb{E}[\theta \mid Y]$ is the optimal summary statistic under squared-error loss and learn S with deep networks, the result underlying the sufficient-statistic discussion of Section 3.2. The cost of the method is that the acceptance rate falls as $\epsilon^{\dim(S)}$, so it becomes prohibitive when the summary dimension is large, and when S is not sufficient the approximation is biased for every ϵ. Approximate Bayesian computation has no differentiable training objective, in contrast to the pinball loss of generative Bayesian computation, the flow log-likelihood, and the evidence lower bound; neural variants (Papamakarios and Murray, 2016; Cranmer et al., 2020) replace the rejection step with a trained density estimator, exhibiting the method as the conceptual ancestor of simulation-based inference.

Fiducial inference

Fiducial inference constructs a data-dependent distribution over parameters without a prior (Hannig et al., 2016). Given a structural model $Y = G_{\mathrm{fwd}}(U, \theta)$, in which G_{fwd} maps latent noise and parameter to data (the opposite direction from the inverse posterior map $\theta = G(Y, Z)$) and $U \sim F_U$ is known, invertibility in θ for fixed U yields a fiducial map $\theta = Q_Y(U) := G^{-1}_{\mathrm{fwd}}(Y, U)$, exactly the parameter–outcome inverse map of Section 3; propagating the randomness of U induces a distribution over θ. For the additive model

$Y = \theta + U$ with $U \sim \mathcal{N}(0,1)$ the fiducial distribution is $\theta \mid Y = y \sim \mathcal{N}(y,1)$. For a sample $Y_1, \ldots, Y_n \sim \mathcal{N}(\mu, \sigma^2)$, the sufficient statistics satisfy $\bar{Y} = \mu + \sigma V_1$ and $S^2 = \sigma^2 V_2$, where $S^2 = \frac{1}{n-1}\sum_{i=1}^{n}(Y_i - \bar{Y})^2$ and the independent latents are $V_1 \sim \mathcal{N}(0, 1/n)$ and $V_2 \sim \text{Gamma}(\frac{n-1}{2}, \frac{n-1}{2})$; inverting these relations ($\mu = \bar{Y} - \sigma V_1$, $\sigma = S/\sqrt{V_2}$) gives the fiducial distributions for μ and σ. A fiducial rejection sampler, for observed data $Y = y$ and draws $i = 1, \ldots, N$, samples $U_i^\star \sim F_U$, solves $\theta_i^\star = \arg\min_\theta \|y - G_{\text{fwd}}(U_i^\star, \theta)\|$, and accepts $\theta_i^\star$ when $\|y - G_{\text{fwd}}(U_i^\star, \theta_i^\star)\| \leq \epsilon$, a procedure structurally identical to approximate Bayesian computation: both simulate auxiliary randomness, invert the forward model, and accept on a distance criterion. The distinction is conceptual, since fiducial inference is prior-free while approximate Bayesian computation targets the Bayesian posterior; in regular models the fiducial distribution coincides with the posterior under Jeffreys' prior, the Jacobian of G_{fwd}^{-1} encoding the Fisher information, and outside such models the generalized fiducial framework of Hannig et al. (2016) supplies the additional conventions. Quantile-based generative Bayesian computation, by contrast, requires neither a structural equation nor invertibility, approximating $F^{-1}_{\theta|y}(\tau)$ from simulated pairs directly.

Normalizing flows and independent component analysis

Normalizing flows and independent component analysis (ICA) share a common idea, modeling observed data as an invertible transformation of independent latent variables, with ICA the linear special case. In ICA (MacKay, 1999), the data $X \in \mathbb{R}^K$ is a linear mixture $X = AZ$ of independent latent sources $Z \in \mathbb{R}^K$, and the goal is to recover the matrix $W = A^{-1}$ so that $Z = WX$ (Figure 4). The sources are mutually independent, $p_Z(z) = \prod_{k=1}^{K} p_k(z_k)$, with each p_k non-Gaussian, since a Gaussian Z would make the model rotationally invariant and hence non-identifiable. Writing the joint density as $p(x, z \mid A) = \delta(x - Az)\, p_Z(z)$ and marginalizing the latent variable gives the closed-form likelihood

$$p_X(x \mid A) = \int p(x, z \mid A)\, dz = \int \delta(x - Az)\, p_Z(z)\, dz = \frac{1}{|\det A|} \prod_{k=1}^{K} p_k\big((A^{-1}x)_k\big),$$

using the Dirac scaling identity $\int \delta(x - Az) f(z)\, dz = |\det A|^{-1} f(A^{-1}x)$, valid for invertible A. ICA is thus a latent-variable model with a deterministic linear decoder $X = AZ$ and an independent latent prior.

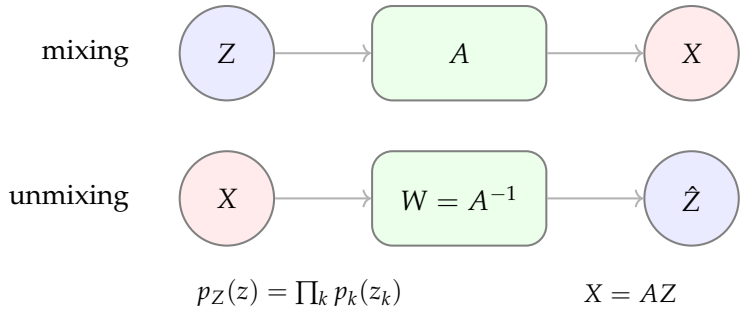

Figure 4: Linear independent component analysis. The mixing matrix A maps independent latent sources Z to observed data X; inference recovers the unmixing matrix $W = A^{-1}$.

Normalizing flows generalize this from linear to nonlinear, learnable maps (Figure 5). Let $X \in \mathbb{R}^D$ have unknown density p_X, modeled as $X = G(Z)$ for a latent $Z \sim p_Z$ and an invertible G with tractable Jacobian, so that the change-of-variables formula gives exact densities,

$$p_X(x) = p_Z(z) \left|\det J_G(z)\right|^{-1}, \qquad z = G^{-1}(x), \tag{5}$$

with $J_G(z) = \partial G(z)/\partial z$. To make G both flexible and tractable it is composed of L simpler bijections, $G = T_L \circ \cdots \circ T_1$, with forward pass $z_k = T_k(z_{k-1})$ and inverse $z_{k-1} = T_k^{-1}(z_k)$; the Jacobian determinant factorizes by the chain rule, $\det J_G(z) = \prod_{k=1}^{L} \det\big(\partial T_k(z_{k-1})/\partial z_{k-1}\big)$, so the log-likelihood of observed data is

$$\log p_X(x) = \log p_Z(G^{-1}(x)) - \log\left|\det J_G(G^{-1}(x))\right|, \tag{6}$$

maximized to fit G. Common building blocks are affine couplings and invertible linear maps; the framework originates with Dinh et al. (2015) and the Real NVP model (Dinh et al., 2017) and has been applied to conditional density estimation (Trippe and Turner, 2018) and posterior approximation (Rezende and Mohamed, 2015a; Müller et al., 2019). An invertible neural network (Dinh et al., 2017) is such a bijection with tractable Jacobian: triangular constructions give efficient determinants (Song et al., 2019), common architectures can have unstable inverses (Behrmann et al., 2021), and HINT networks (Kruse et al., 2021) transport a complex posterior toward a simple base measure for direct posterior sampling.

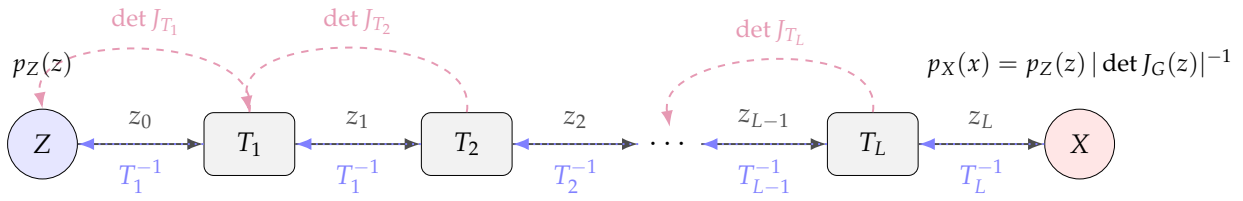

Figure 5: Normalizing flow. A composition of invertible maps $T_1, \ldots, T_L$ transports a simple base density p_Z to the data density p_X; the inverse pass and the factorized Jacobian determinant give exact likelihoods.

In the parameter–outcome class a conditional flow $f_\phi(\cdot \mid Y)$ (with ϕ denoting network parameters) learns the bijection $\theta = f_\phi^{-1}(Z \mid Y)$ from a base law $Z \sim p_Z$ to the posterior, with the explicit objective

$$\max_\phi \frac{1}{N} \sum_{i=1}^{N} \Big[\log p_Z\big(f_\phi(\theta_i \mid Y_i)\big) + \log \big|\det J_{f_\phi}(\theta_i \mid Y_i)\big| \Big]$$

over simulated pairs $\{(\theta_i, Y_i)\}_{i=1}^N$, giving the exact approximate-posterior density $\hat{p}(\theta \mid Y) = p_Z(f_\phi(\theta \mid Y))\,|\det J_{f_\phi}(\theta \mid Y)|$, an advantage over the quantile parameterization of generative Bayesian computation, which returns quantiles but no closed-form density (Papamakarios et al., 2019; Greenberg et al., 2019). The invertibility requirement forces $\dim(\theta) = \dim(Z)$ (or padding) and, for posteriors with well-separated modes, a continuous path between modes through the base law that can distort their shapes, a constraint that quantile and diffusion estimators avoid. The same construction serves the other two classes by changing the conditioning: an unconditional flow $X = G(Z)$ is an outcome generator whose exact likelihood scores draws (used, for example, to flag low-likelihood network traffic as anomalous), and a covariate-conditioned flow $Y = G(X, Z)$ is a predictive generator with exact conditional density.

4.3 Generators for Predictive Distributions

The third generator class targets prediction with uncertainty quantification. The data object is an input–output sample or simulator table

$$\{(X_i, Y_i)\}_{i=1}^N,$$

and the target is the conditional outcome distribution

$$Y \mid X = x \overset{d}{=} G(x, Z), \qquad Z \sim \text{Unif}(0, 1).$$

This is a forward generator rather than an inverse generator: the input is a covariate vector, design point, parameter setting, or simulator configuration, and the output is a predictive distribution for future outcomes.

Classical regression, Gaussian process emulation (Kennedy and O'Hagan, 2001), conditional flows, conditional variational autoencoders, conditional adversarial generators, and conditional diffusion models all fit within this class when their output is a conditional distribution for $Y \mid X$. Their differences are operational. Gaussian process emulators give smooth interpolation and uncertainty through covariance modeling, but require specifying covariance structure and can scale poorly. Conditional flows provide exact density evaluation if an invertible map is appropriate. Conditional variational autoencoders and adversarial generators produce fast conditional samples but require empirical calibration checks. Conditional diffusion models are strong for complex multimodal predictive distributions, but sampling remains sequential and expensive.

Quantile neural networks are direct predictive generators when the target is a set of conditional quantiles or prediction intervals. Exchanging the roles of θ and Y in generative Bayesian computation detailed in the next section gives the forward map $Y = G(X, Z)$. This role reversal is important: the same algorithm can estimate a posterior generator from (θ, Y) pairs or a predictive generator from (X, Y) pairs, but the statistical interpretation changes from inverse inference to forward prediction. Conformalized quantile regression (Romano et al., 2019) augments such a quantile generator with a distribution-free calibration step, delivering finite-sample marginal coverage (Vovk et al., 2005; Lei et al., 2018). Engression (Shen and Meinshausen, 2024) is a closely related distributional-regression generator that targets the conditional law through a different training objective.

Another example is conditional normalizing flow, detailed in the parameter–outcome subsection above, which gives full predictive distributions for time-to-event outcomes. With survival time $Y \in \mathbb{R}_+$ and covariates $X \in \mathbb{R}^d$ (age, treatment indicator, biomarkers), the flow parameterizes $Z = f_\phi(Y \mid X)$, mapping survival times to a base law $Z \sim \mathcal{N}(0, 1)$ conditional on covariates, with conditional density

$$p(y \mid x) = p_Z\big(f_\phi(y \mid x)\big) \left|\partial f_\phi(y \mid x)/\partial y\right|,$$

maximized over observed pairs, with censoring handled by integrating the density over

the censored region. The conditional quantile function follows directly as $F^{-1}_{Y|X}(\tau) = f^{-1}_{\phi}(\Phi^{-1}(\tau) \mid X)$, giving median survival, interquartile intervals, and any quantile from a single fit. This is the predictive construction $Y = G(X, Z)$ of this class with exact density. Compared with the Cox proportional-hazards model (Cox, 1972), which models the hazard parametrically and provides no direct density, the conditional flow is nonparametric in the covariate–outcome relationship and supplies the full conditional density and all quantiles, at the cost of the invertibility constraint and of training the flow rather than fitting a partial likelihood.

5 Generative Bayesian Computation

This section presents generative Bayesian computation, a framework for the second generator class: posterior generation from simulated parameter–outcome pairs. The method learns the inverse posterior map directly using quantile-based deep neural networks. The same quantile-network architecture, applied with the roles of θ and Y exchanged, yields a forward emulator $Y = G(X, \tau)$ for predictive inference; this dual construction is used in Section 6.

The framework is applicable whether or not a tractable likelihood is available. When a forward model $Y = f(\theta)$ is available but the likelihood $p(Y \mid \theta)$ is intractable, or when Markov chain Monte Carlo posterior sampling is too costly, generative Bayesian computation bypasses density evaluation and iterative sampling entirely by relying on forward simulation alone. The central object is the inverse posterior map in conditional form

$$\theta = F^{-1}_{\theta|Y=y}(\tau), \qquad \tau \sim \mathrm{Unif}(0, 1),$$

which we will write equivalently as $\theta = F^{-1}_{\theta|y}(\tau)$ when the conditioning variable is unambiguous. For scalar θ, this is the conditional quantile function introduced in Section 3.3.

An optimal-transport objective for vector-valued parameters

For vector-valued θ the scalar inverse cumulative distribution function is no longer unique. Carlier et al. (2016) give a coordinate-free generalization based on the optimal-transport map of Brenier (1991), which characterizes a vector quantile function through the gradient of a convex potential. We instead adopt an autoregressive factorization that fixes a coordinate ordering and uses scalar conditional quantiles at each stage:

$$\theta = \left(F^{-1}_{\theta_1|Y=y}(\tau_1),\ F^{-1}_{\theta_2|\theta_1,Y=y}(\tau_2),\ \ldots,\ F^{-1}_{\theta_d|\theta_1,\ldots,\theta_{d-1},Y=y}(\tau_d) \right),$$

where each conditional quantile is evaluated at the observed data value $Y = y$ and, after the first coordinate, at the previously generated parameter coordinates. Two properties of this choice should be stated plainly. First, it is *order-dependent*: the factorization, and hence the resulting joint generator, depends on the coordinate ordering. One-dimensional marginal quantiles of each coordinate are well defined, but joint credible regions and the shape of marginal intervals are not invariant to the ordering, and a practitioner reporting such summaries should fix and disclose the ordering. The construction of Brenier (1991) avoids this at the cost of solving an optimal-transport problem. Second, *approximation error propagates*: each stage conditions on previously generated (hence imperfect) coordinates, so errors compound along the chain. We are not aware of a sharp rate analysis for the d-dimensional autoregressive estimator (the rate stated below is for the scalar map), and we emphasize both the ordering dependence and the error propagation as open issues. [2]

The Brenier route can be turned into a trainable objective that keeps the deterministic-transport structure of the scalar quantile map. By Brenier's theorem, for an absolutely continuous baseline μ the optimal transport map carrying μ onto the posterior $\pi(\cdot \mid y)$ under quadratic cost is the gradient of a convex potential (Brenier, 1991; McCann, 1995; Villani, 2009),

$$\theta = \nabla\psi(U, y), \qquad \psi(\cdot, y) \text{ convex}, \qquad (\nabla\psi)_{\#}\,\mu = \pi(\cdot \mid y), \quad U \sim \mu,$$

and the map is μ-almost-everywhere unique, so the latent U is identified exactly as the probability level τ is in one dimension. The scalar check loss no longer applies, but a strictly proper scoring rule takes its place: the energy score (Székely and Rizzo, 2013; Gneiting and Raftery, 2007)

$$S(P, \theta) = \mathbb{E}_{Z\sim P}\|Z - \theta\| - \tfrac{1}{2}\,\mathbb{E}_{Z,Z'\sim P}\|Z - Z'\|,$$

with Z, Z' independent draws from P, has expected value uniquely minimized at $P = \pi(\cdot \mid y)$, so fitting $\nabla\psi$ against it is consistent for the posterior. This preserves the three defining features of generative Bayesian computation, deterministic transport, a fixed proper objective, and the absence of a discriminator, in dimension greater than one, at the cost of solving an optimal-transport problem in place of a sequence of scalar quantile

[2] Each conditional quantile is parameterized by a neural network trained on simulated pairs $\{(\theta_i, Y_i)\}_{i=1}^{N}$. The training objective is the pinball loss defined in Section 3.3; Appendix B describes the quantile-theoretic foundations. An open-source implementation of the generative Bayesian computation framework is available in the `gbc` Python package.`https://github.com/VadimSokolov/gbc`.

regressions.

Neural-network estimation of full quantile functions was popularized in distributional reinforcement learning, where Dabney et al. (2017) represent the distribution of future rewards rather than only its mean. We use that literature only for its numerical mechanism: embedding a quantile index and training the network with the pinball loss. In the present paper the conditioning variable is the observed data Y, the target is the Bayesian parameter θ, and no reinforcement-learning objective or utility function is used.

Quantile neural networks

The posterior distribution can be represented through its inverse cumulative distribution function, following the classical probability integral transform and the quantile-function viewpoint developed by Parzen (2004), who frames quantile methods as direct alternatives to density estimation:

$$\theta = F^{-1}_{\theta|Y}(\tau), \quad \tau \sim \text{Unif}(0,1).$$

More generally, when a sufficient statistic $S(Y)$ is used, this fits within the inverse map framework $\theta = G(S(Y), \tau)$, where the posterior is characterized by

$$\theta \overset{d}{=} G(S(Y), \tau), \quad \tau \sim \text{Unif}(0,1).$$

A deep neural network parameterizes G, taking both the data summary $S(Y)$ and the quantile level τ as inputs and producing the corresponding posterior quantile as output. The network is trained by minimizing the pinball loss (Section 3.3) over simulated pairs $\{(\theta_i, Y_i)\}_{i=1}^N$. The Kolmogorov–Arnold representation theorem (Kolmogorov, 1957) guarantees that any multivariate continuous function can be expressed as a superposition of univariate functions; Schmidt-Hieber (2020) establish convergence rates for deep rectified-linear-unit networks approximating such functions. By exchanging the roles of θ and Y, the same construction gives a forward emulator $Y = G(X, \tau)$ for predictive inference (Section 6), with identical network architecture and training objective.

Quantile neural networks differ from normalizing flows. Flows require an invertible architecture and produce exact densities via the change-of-variables formula (Rezende and Mohamed, 2015b; Dinh et al., 2015, 2017; Bond-Taylor et al., 2022), but this constraint limits architectural flexibility. Quantile networks impose no invertibility requirement: they directly approximate the inverse cumulative distribution function without modeling the forward density. This makes them applicable when the parameter space is lower-

dimensional than the data space (a common setting in Bayesian inference) and allows the use of standard feedforward architectures (Kingma and Welling, 2022). The trade-off is that quantile networks do not provide a closed-form density for the approximate posterior.

Convergence rates

The approximation quality of the generative Bayesian computation quantile estimator is governed by the smoothness of the posterior quantile function $F^{-1}_{\theta|y}(\tau)$ and the number of simulated training pairs N. Suppose the posterior quantile function, viewed as a map $(\tau, y) \mapsto F^{-1}_{\theta|y}(\tau)$, is a composite function with L layers, where the ℓ-th layer acts on d_ℓ variables with Hölder smoothness index β_ℓ. The effective dimension at each layer is d_ℓ, and the effective smoothness after composition is $\beta^*_\ell = \beta_\ell \prod_{k=\ell+1}^{L} \min(\beta_k, 1)$. Under these conditions, deep rectified-linear-unit network estimators of the posterior quantile function achieve the convergence rate

$$\mathbb{E}\left[\|\hat{F}^{-1}_N - F^{-1}_{\theta|y}\|^2\right] = O\left(\max_{1\le\ell\le L} N^{-2\beta^*_\ell/(2\beta^*_\ell+d_\ell)}\right),$$

where the expectation is over the simulated training pairs. The conditional-quantile-regression form of this rate is due to Padilla et al. (2022); the multi-quantile training case is treated by Moon et al. (2021); both rest on the composition theory of Schmidt-Hieber (2020). See Appendix A for the Bayes risk framework underlying these bounds. Two caveats are important. First, this is an approximation/statistical rate for the *estimator class*; it excludes optimization error and is not specific to the cosine-embedding architecture used below, so the trained network's error may exceed it. Second, and more important for interpretation, the rate controls the distance between $\hat{F}^{-1}_N$ and the *exact posterior* quantile function, not the distance to the data-generating truth. The total error a practitioner incurs decomposes as

$$\underbrace{\left(\text{exact posterior inferential error}\right)}_{\text{governed by data size } n} + \underbrace{\left(\hat{F}^{-1}_N - \text{exact posterior}\right)}_{\text{governed by simulation budget } N}.$$

Only the second term is N-controlled; the first is the usual posterior concentration, driven by the actual data and unaffected by N. The statement that "N is the binding constraint" refers to the second term alone. Within that term, the composite structure of the posterior quantile map also shows how generative Bayesian computation can mitigate the curse of dimensionality: if the posterior concentrates on a low-dimensional sufficient statistic

$S(Y)$ of intrinsic dimension $d_* \ll \dim(Y)$, the effective dimension in the rate is d_* rather than the ambient data dimension, just as deep networks exploit low-dimensional composition structures (Schmidt-Hieber, 2020). This is the formal sense in which learning $S(Y)$ improves statistical properties of generative Bayesian computation. Analogous closed-form rates under the same conditional-quantile risk framework are not directly available for the variational autoencoder evidence-lower-bound objective or for the generative adversarial network objective, since those objectives do not directly control the L^2 approximation error of the posterior quantile function; the use of the pinball loss in generative Bayesian computation (which directly targets the conditional quantile function) is what makes this rate analysis tractable (Koenker and Bassett Jr, 1978).

Implicit quantile networks for posterior estimation

Dabney et al. (2018) introduce an implicit quantile network, a deep learning architecture that approximates the full quantile function of a conditional distribution. In our setting, the network estimates $F^{-1}(\tau, Y)$, the τ-quantile of θ given data Y, where $\tau \in (0,1)$ is a quantile index. Sampling from the approximate posterior amounts to drawing $\tau \sim \text{Unif}(0,1)$ and evaluating the network at (τ, Y).

In the generative Bayesian computation framework, we parameterize the posterior quantile function as a composite network $F^{-1}(\tau, Y) = f(\tau, Y; w)$ (Figure 6). Following Dabney et al. (2018), this function is decomposed as:

$$F^{-1}(\tau, Y) = f(\tau, Y; w) = g\left(\psi(Y) \circ \chi(\tau)\right),$$

where g, ψ, and χ are feedforward neural networks, $\circ$ denotes element-wise (Hadamard) multiplication, and w collects all network weights. The quantile embedding $\chi(\tau)$ is constructed using a cosine basis followed by a rectified-linear-unit (ReLU) activation:

$$\chi_j(\tau) = \text{ReLU}\left(\sum_{k=0}^{p-1} \cos(\pi k \tau) w_{kj} + b_j\right),$$

providing a smooth encoding of the quantile index that allows the network to share information across nearby quantile levels. The network is estimated by minimizing the pinball loss over a randomly drawn set of quantile levels $\tau \sim \text{Unif}(0,1)$ and simulated pairs $\{(\theta_i, Y_i)\}_{i=1}^N$. This architecture can estimate heteroskedastic and non-Gaussian posteriors without explicit likelihood specification, though it requires a sufficiently large training sample to control approximation error.

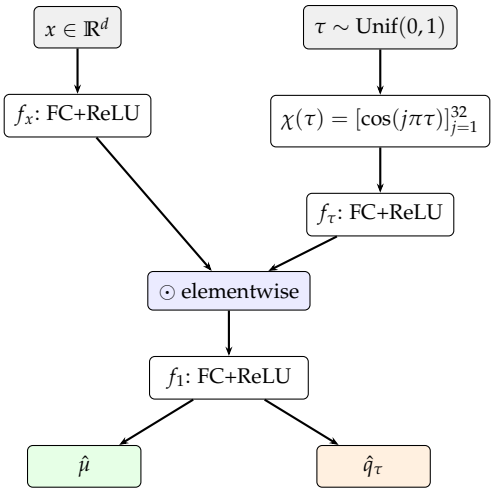

Figure 6: Implicit quantile network used by generative Bayesian computation. The conditioning input x (the data summary $S(Y)$ for posterior inference, or covariates for prediction) and the quantile level τ are embedded separately, the latter through a cosine feature map $\chi(\tau)$, then combined by an elementwise (Hadamard) product and decoded to the estimated τ-quantile $\hat{q}_\tau$ and a conditional-mean $\hat{\mu}$. The branches f_x, f_τ, and f_1 correspond to ψ, χ, and g above; the mean head supports the composite training loss.

Generative Bayesian computation and the adversarial encoder–decoder both push a simple baseline measure forward onto the target conditional law, but they differ in what map is learned and what objective is optimized, and the differences favor the quantile-transport construction for inference. Appendix E develops the contrast point by point: a convex risk versus a saddle point, gradient quality, consistency, latent identifiability, density and cumulative-distribution-function access, as well as conditioning and amortization. Table 6 (in that appendix) summarizes the six contrasts side by side, and the closing caveats paragraph there extends the comparison to the multivariate setting of Brenier (1991).

An open-source implementation of the generative Bayesian computation framework is available in the `gbc` Python package (`https://github.com/VadimSokolov/gbc`). The Ebola application of Section 6.1 uses a feedforward implicit-quantile architecture in the style of Dabney et al. (2018): a quantile embedding $\chi(\tau)$ formed from a cosine basis of dimension p followed by a rectified-linear-unit activation, combined with a data branch $\psi(Y)$ via Hadamard product, and decoded by a feedforward head g. The composite training objective $\mathcal{L} = \mathcal{L}_{\text{pinball}} + \lambda_1 \mathcal{L}_{\text{mean}} + \lambda_2 \mathcal{L}_{\text{mono}}$ combines the pinball loss with a conditional-mean loss and a monotonicity penalty against quantile crossings; the weights $\lambda_1, \lambda_2 > 0$ are selected by cross-validation. Training uses Adam with a piecewise-constant learning-rate schedule, and the dimensionality reduction step of Section 6.1 uses partial least squares fit on the simulated (θ, Y) pairs. The Fadikar et al. (2018) Ebola simulator

output ($m = 100$ design points, 100 stochastic replicates each, $N = 10{,}000$ trajectories of length 56) and the code used to produce all figures and coverage diagnostics in Section 6 are distributed with the `gbc` package.

6 Applications

This section mirrors the three generator classes of Section 4. The first application is a parameter–outcome generator for inverse inference in an Ebola agent-based epidemic model. The second is a predictive generator for forward emulation of the same Ebola simulator. The third application is a counterfactual outcome generator with a binary treatment, discussed in Appendix F.

6.1 Parameter–outcome generator: Ebola inverse inference

The first application uses the multi-output agent-based epidemic model documented in Fadikar et al. (2018). Here the training object is a simulated parameter–outcome table, and the target is the posterior distribution over static simulator inputs given an observed epidemic trajectory. This is the parameter–outcome generator class of Section 4.

After the 2014-2015 West Africa Ebola outbreak, the Research and Policy for Infectious Disease Dynamics program at the National Institutes of Health convened a workshop to compare forecasting approaches used during the outbreak. A stochastic agent-based model (Ajelli et al., 2018) generated synthetic populations with varying degrees of data accuracy, availability, and intervention measures. Transmission by an infected individual is determined probabilistically based on contact duration and $d = 5$ static inputs $(\theta_1, \ldots, \theta_5)$ described in Table 1.

Parameter	Description	Range
θ_1	Transmission probability	$[3 \times 10^{-5}, 8 \times 10^{-5}]$
θ_2	Initial infected count	$[1, 20]$
θ_3	Hospital intervention delay	$[2, 10]$
θ_4	Hospital intervention efficacy	$[0.1, 0.8]$
θ_5	Travel reduction from intervention	$[0, 2]$

Table 1: Static inputs for the Ebola agent-based model.

A single simulator run outputs cumulative infections over 56 weeks. We use the data set of Fadikar et al. (2018): $m = 100$ parameter settings generated through a space-filling

symmetric Latin hypercube design, with 100 stochastic replicates per setting, for a total of $N = 10{,}000$ simulated epidemic trajectories. The log trajectories are shown in Figure 7.

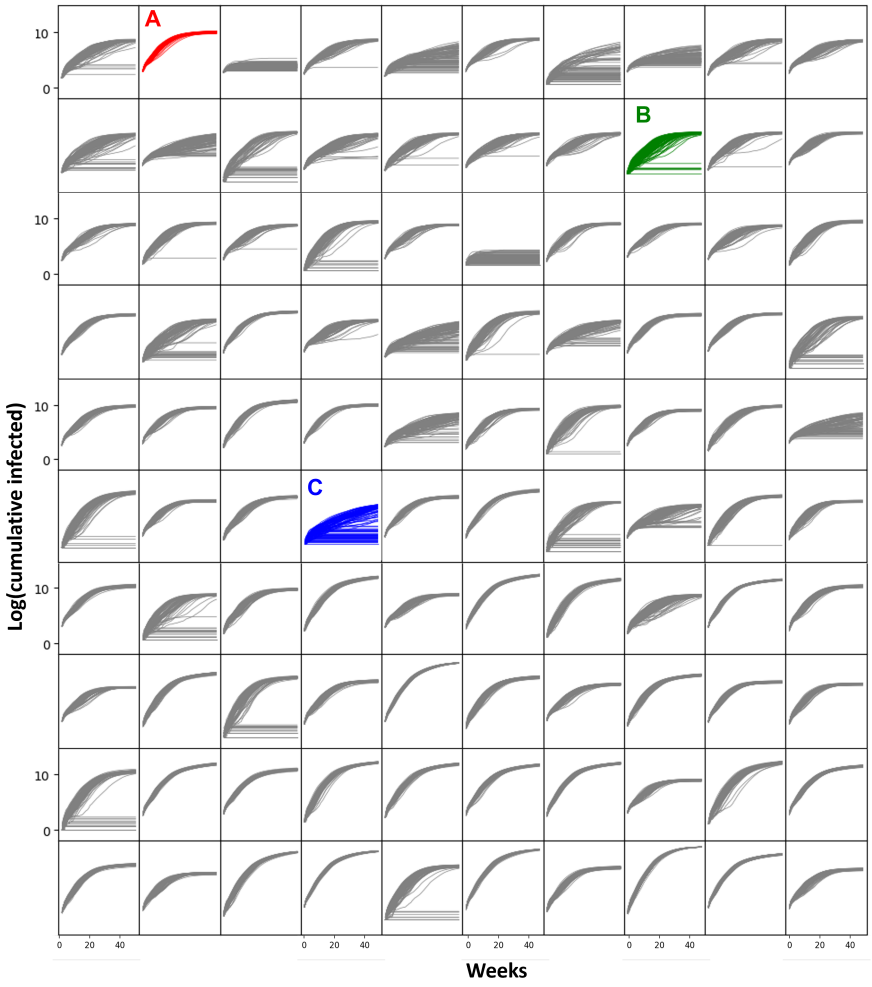

Figure 7: Simulated trajectories of cumulative disease incidence across 56 weeks for all $m = 100$ parameter settings, with 100 stochastic replicates per setting (gray lines). The three parameter settings used as holdouts in the forward-emulation application of Section 6.2, labeled A, B, and C, are highlighted in red, green, and blue respectively.

Given an observed trajectory Y, the inverse-inference target is $p(\theta \mid Y)$. We train a quantile neural network on simulated pairs $\{(\theta_i, Y_i)\}_{i=1}^{N}$ using partial least squares for dimensionality reduction of the 56-dimensional output. The network is trained with a composite loss

$$\mathcal{L} = \mathcal{L}_{\text{pinball}} + \lambda_1 \mathcal{L}_{\text{mean}} + \lambda_2 \mathcal{L}_{\text{mono}},$$

where $\mathcal{L}_{\text{pinball}}$ is the pinball loss of Koenker and Bassett Jr (1978) targeting the conditional quantile function, $\mathcal{L}_{\text{mean}}$ is an L_1 loss on the conditional mean (stabilizing training), and $\mathcal{L}_{\text{mono}}$ is a monotonicity regularizer penalizing quantile crossings; the regularization

weights $\lambda_1, \lambda_2 > 0$ are selected by cross-validation. The network architecture follows Dabney et al. (2018).

The generative Bayesian computation model achieves empirical 90% marginal per-coordinate credible interval coverage of 89% (θ_1), 88% (θ_2), 90% (θ_3), 90% (θ_4), and 90% (θ_5) on 2,000 held-out test observations (an 80/20 split of the $N = 10{,}000$ simulated trajectories). All five marginal coverages are within 2 percentage points of the nominal 90% level (standard error $\approx 0.7\%$ at $n = 2{,}000$). The intervals are formed independently for each coordinate from the network's 0.05 and 0.95 quantile outputs and do not certify joint posterior coverage. Figure 8 shows posterior medians and intervals for 30 randomly selected held-out observations. Parameter θ_5 is recovered accurately, while θ_3 and θ_4 exhibit greater posterior uncertainty consistent with weaker identifiability in the simulator output; the marginal coverage is nevertheless close to nominal. The `gbc` package provides post-hoc conformal calibration when stricter coverage guarantees are required, and Section G below probes the calibration further with simulation-based-calibration rank histograms that reveal structural mismatches the marginal coverage statistic alone does not detect.

The coverage statistics reported above measure frequentist coverage averaged over the simulated joint (θ, Y) in the held-out split. In the language of Talts et al. (2018), this is simulation-based calibration of the *average* posterior, and it is the standard diagnostic for amortized neural inference. It is not the same as a check that the approximation $\hat{p}(\theta \mid Y = y)$ agrees with the exact posterior $p(\theta \mid Y = y)$ at any particular observed y; calibration averages can mask systematic errors at specific y, including under- or over-dispersion that cancels across the test set. Moreover, the pinball-loss training criterion controls the conditional quantile of each coordinate of θ marginally, but does not by itself certify a coherent joint posterior over θ once the autoregressive factorization of Section 5 is invoked. Two complementary checks address this. First, rank histograms in the sense of Talts et al. (2018) test uniformity of the simulation-based calibration ranks per coordinate. Second, where a closed-form posterior is available (a conjugate model is a natural choice) one can overlay generative Bayesian computation samples directly against the exact posterior to test agreement at specific y. These two checks are complementary to the coverage statistics reported above and should accompany them in routine practice.

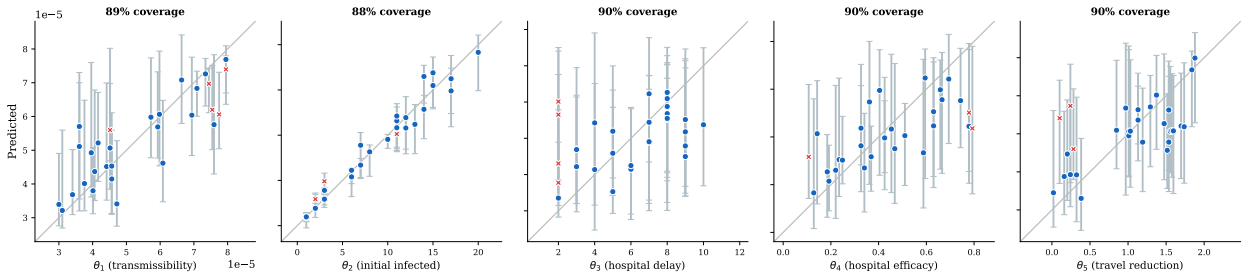

Figure 8: Predicted versus true parameter values for 30 randomly selected observations from the 2,000-observation held-out test set. Each panel shows one model parameter θ_j: the posterior median (dots) against the true value, with 90% credible intervals (vertical bars) from the network's 0.05 and 0.95 quantile outputs. Blue dots indicate the true value falls within the interval; red dots indicate miscoverage. Coverage rates shown are empirical marginal per-coordinate rates computed over the full 2,000-observation test set; the plotted points are a 30-observation display sample.

For a single held-out observation, Figure 9 overlays the generative Bayesian computation posterior samples on a nearest-neighbor reference distribution drawn from the closest training trajectories, providing a second view of the per-coordinate fit on one fixed y rather than the cross-observation summary of Figure 8.

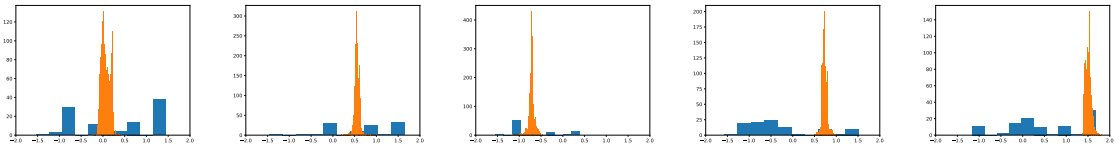

Figure 9: Approximate posterior histograms for observation $y^{(7080)}$. Each panel overlays the generative Bayesian computation posterior samples (orange; generated by drawing $\tau \sim \text{Unif}(0, 1)$ and evaluating the trained quantile network) with a nearest-neighbor reference distribution (blue; the parameter values from the 100 agent-based model simulations whose output trajectories are closest to $y^{(7080)}$). The reference distribution is an approximate posterior proxy and may be overdispersed due to the limited number of neighbors and imperfect matching in output space.

The relevant baseline is the quantile kriging approach of Fadikar et al. (2018). It fits a Gaussian process emulator on the augmented input (θ, τ) using the model-assisted calibration framework of Higdon et al. (2008). Their approach requires specifying covariance structure, basis functions, and nugget parameters, with calibration performed by Markov chain Monte Carlo. Both approaches use the same 10,000 simulator runs. After training, the quantile network produces each posterior draw by one network evaluation, whereas

Gaussian-process calibration requires a separate Markov chain Monte Carlo calibration step. Approximate Bayesian computation would require repeated nearest-neighbor or rejection calculations for each observation, and MCMC with simulated likelihood would call the simulator $O(N_{\text{chain}})$ times per observation. Appendix D tabulates these costs and their trade-offs.

Appendix G reports three further calibration checks of the generative Bayesian computation posterior on the Ebola inverse problem: a closed-form conjugate benchmark where the exact posterior is available (Kolmogorov–Smirnov statistic between 0.032 and 0.040 against the exact Beta), per-parameter simulation-based calibration rank histograms (the rank distribution rejects uniformity for θ_2–θ_5 even though marginal coverage is near-nominal, so the two diagnostics are complementary), and a detailed comparison against a flow-based neural posterior estimator on the same simulated table (the quantile network has 4–19 percentage points higher marginal coverage than a masked autoregressive flow at default settings).

6.2 Predictive generator: Ebola forward emulation

The second application uses the same Ebola simulator but changes the generator class. The input is now a parameter setting, and the target is the predictive distribution of the 56-week epidemic trajectory. This is a forward prediction problem, not inverse inference. The relevant training object is an input–output table indexed by the simulator input and a quantile level.

Different Ebola parameter settings produce strongly divergent behaviors: some trajectories follow a mean trend while others exhibit bimodality, heteroskedasticity, or remain flat. To handle this stochastic variability, Fadikar et al. (2018) replace the 100 replicates at each parameter setting with $n_\tau = 5$ quantile-based summary trajectories at levels $\tau \in \{0.05, 0.275, 0.5, 0.725, 0.95\}$. This reduces the stochastic output to a deterministic function of the augmented input

$$\tilde{\theta} = [\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \tau]\,.$$

For testing, three parameter settings, labeled A, B, and C, and their respective $n = 100$ simulated outputs were excluded from the training set. Figure 10 shows the five empirical quantiles of these holdout scenarios.

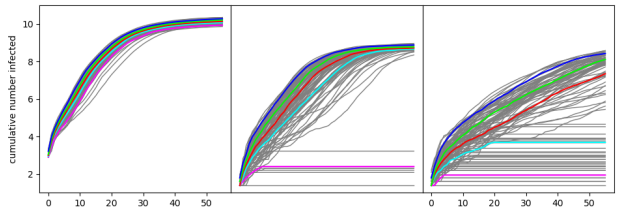

Figure 10: For each holdout scenario, labeled A, B, and C, the 100 simulated trajectories of cumulative disease incidence over 56 weeks are shown as gray lines. The five colored lines represent the $[0.05, 0.275, 0.5, 0.725, 0.95]$ quantiles, indexed by the additional input τ.

Figure 11 shows results from a deep neural network combined with a Gaussian process emulator. The neural network learns a nonlinear reduction of the 56-dimensional output into a 3-dimensional latent score, and independent Gaussian process regressions are then fitted to the latent scores. This is a predictive generator: the output is a conditional distribution over future trajectories given a parameter setting and quantile index. It is closest to the prediction class in Section 4, not to a normalizing flow, because the reduction is not constrained to be invertible and no exact density is evaluated.

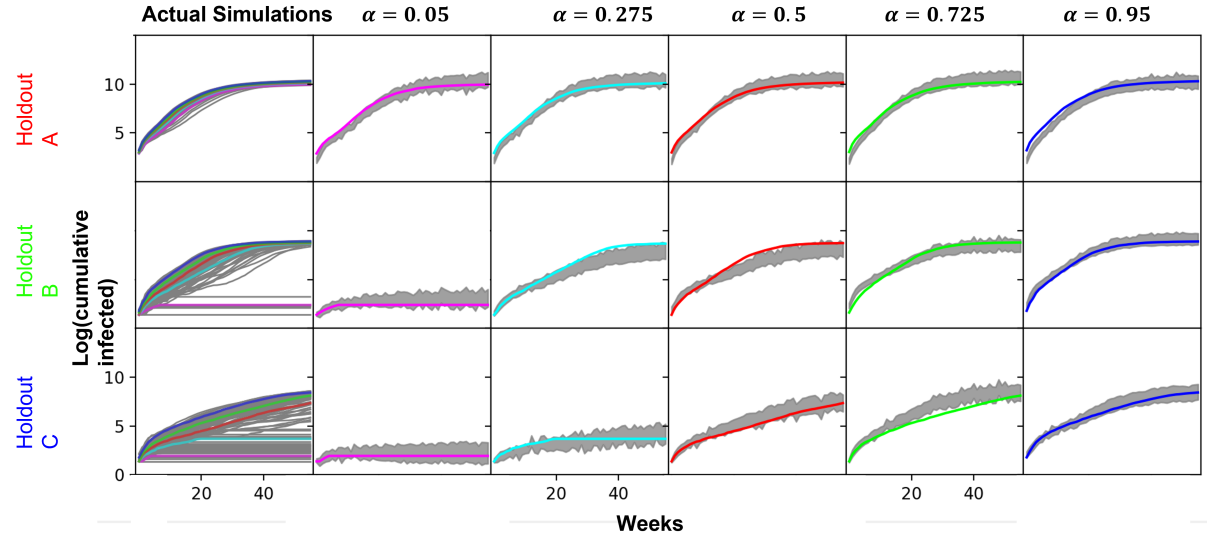

Figure 11: Forward emulation with a deep neural network and Gaussian process model. The first column shows the 100 agent-based model replicates and their empirical quantiles for each holdout scenario. The remaining five columns show the model's predicted median (colored line) and 90% prediction interval (gray band) for each quantile level τ.

A visual inspection of the lower quantiles ($\tau = 0.05, 0.275$) suggests potential improve-

ment in lower-quantile calibration relative to the quantile kriging approach of Fadikar et al. (2018), with the true quantile trajectory remaining inside the 90% predictive band across the full 56-week timeline for several cases. This is a qualitative observation on a single application, not a formal claim: a quantitative comparison via root mean squared error or coverage on a held-out evaluation set would be needed to confirm or refute the impression, and we have not conducted such a comparison. The key conceptual distinction from the previous subsection is that this generator maps inputs forward to outcomes, whereas the inverse-inference generator maps observed outcomes back to plausible parameters.

7 Discussion

We have framed generative Bayesian computation within three generator classes. These are counterfactual outcome distributions, posterior distributions learned from simulated parameter and outcome pairs, and predictive distributions. The unifying idea, formalized through the Kallenberg noise outsourcing theorem, is that all three classes can be recast as learning a deterministic function of inputs and independent reference noise. In compact form, $Y = G(c, Z)$ for outcome generation, $\theta = G(Y, Z)$ for parameter generation, and $Y = G(X, Z)$ for prediction.

The main organizational point is that methods should be evaluated by the generator they are asked to estimate. For counterfactual outcome distributions, the central issue is identification of $Y(a)$ and calibration of the conditional outcome distribution; adversarial, diffusion, variational, flow, and quantile generators are numerical choices after the causal assumptions have been stated. For simulated parameter–outcome pairs, the central issue is inverse inference: the generator must transform an observed outcome into posterior draws, quantiles, or densities for θ. Approximate Bayesian computation, fiducial inference, conditional flows, conditional diffusion, and generative Bayesian computation all belong here when trained on (θ, Y) pairs. For prediction problems, the generator is forward-looking: it maps covariates, design points, or simulator inputs into a predictive distribution for Y.

We present Generative Bayesian computation which is primarily a parameter–outcome generator. It estimates posterior quantiles directly from simulated pairs, avoiding both likelihood evaluation and invertibility constraints. Its role in prediction is obtained by exchanging the roles of parameter and outcome, as in the Ebola forward-emulation application. Its role in counterfactual inference is obtained by changing the training object to causal triples $(\tilde{S}(X), A, Y)$ and interpreting the fitted quantile map as $G(q, \tilde{s}, a)$. The

main trade-off is not the split between neural and non-neural methods. Rejection methods are transparent and simulation-based, but they scale poorly with the dimension of the summary statistic. Flows provide density evaluation, at the cost of invertible maps and tractable Jacobians. Diffusion methods handle multimodality but require sequential sampling. Variational autoencoders and adversarial generators produce fast samples, yet they require calibration checks and may miss tail behavior or modes. Quantile generators directly target conditional quantiles and intervals, but they do not provide a closed-form density. The appropriate choice depends on whether the application requires counterfactual outcome distributions, parameter posteriors, or predictive distributions.

A separate axis of comparison concerns whether the generator is trained once on a fixed simulation budget (amortized inference) or refined adaptively for the specific observation of interest (sequential inference). Amortized methods, including generative Bayesian computation, conditional normalizing flows, and conditional diffusion, fit the generator on a single set of N simulated pairs and produce posteriors for any new observation by direct evaluation; the simulation budget is shared across observations and the per-observation inference cost is one function evaluation. Sequential methods, including sequential neural posterior estimation (Papamakarios and Murray, 2016; Lueckmann et al., 2017; Greenberg et al., 2019), sequential neural likelihood, and sequential neural ratio estimation, focus the simulation budget around the observed datum by drawing new simulations from a proposal that adapts to the current posterior approximation; this can be statistically more efficient per simulator call but discards the amortized property. The choice between the two depends on whether multiple observations are expected (amortized methods are preferred) or only a single observation is of interest with a tight simulator budget (sequential methods may pay off). Appendix D compares these costs explicitly for the Ebola application.

The architectural matrix of Appendix C (Table 3) can be reduced to a short guide to the practical question facing the analyst. If a tractable density of the approximate posterior is required (for likelihood-ratio tests, model selection, or hierarchical embedding), a normalizing flow trained as a neural posterior or likelihood estimator is the natural choice. If only credible intervals and posterior quantiles are needed, a quantile-based generator such as generative Bayesian computation avoids the invertibility constraint and the change-of-variables Jacobian, and is straightforward to calibrate post-hoc by conformal procedures. If the target distribution is multimodal, a diffusion or score-based generator handles modes more gracefully than a single-mode-friendly variational autoencoder. If the parameter or outcome is high-dimensional and a coordinate ordering is unattractive, a Brenier optimal-transport map or a multivariate flow is preferable to the autoregressive

factorization of Section 5. For counterfactual outcome questions the choice of architecture is secondary to the causal assumptions; once those are stated, any conditional generator can serve as the estimator of G. The single highest-leverage point in all three classes is to learn a low-dimensional sufficient summary of the data before fitting the generator: this reduces the effective dimension in the rate of Section 5 and improves both calibration and stability.

Each generator class has distinct unresolved issues. Counterfactual generators inherit the identification challenges of causal inference: unconfoundedness, positivity, and support overlap. Parameter–outcome generators require posterior calibration diagnostics, better handling of high-dimensional parameters, and robustness to simulator misspecification. Predictive generators require conditional calibration under covariate shift and scalable uncertainty quantification for high-dimensional outputs. Across all three classes, the central statistical problem is to connect approximation error of the learned generator to the inferential object the researcher reports.

References

Ajelli, M., Zhang, Q., Sun, K., Merler, S., Fumanelli, L., Chowell, G., Simonsen, L., Viboud, C., and Vespignani, A. (2018). The RAPIDD Ebola forecasting challenge: Model description and synthetic data generation. *Epidemics*, 22:3–12.

Akesson, M., Singh, P., Wrede, F., and Hellander, A. (2021). Convolutional neural networks as summary statistics for approximate bayesian computation. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*.

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

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein generative adversarial networks. In *International Conference on Machine Learning*, pages 214–223. PMLR.

Athey, S., Karlan, D., Palikot, E., and Yuan, Y. (2022). Smiles in profiles: Improving fairness and efficiency using estimates of user preferences in online marketplaces. Technical report, National Bureau of Economic Research.

Bach, F. (2024). High-dimensional analysis of double descent for linear regression with random projections. *SIAM Journal on Mathematics of Data Science*, 6(1):26–50.

Baker, E., Barbillon, P., Fadikar, A., Gramacy, R. B., Herbei, R., Higdon, D., Huang, J., Johnson, L. R., Ma, P., Mondal, A., et al. (2022). Analyzing stochastic computer models: A review with opportunities. *Statistical Science*, 37(1):64–89.

Barron, A. R. (1993). Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information theory*, 39(3):930–945.

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

Behrmann, J., Vicol, P., Wang, K.-C., Grosse, R., and Jacobsen, J.-H. (2021). Understanding and Mitigating Exploding Inverses in Invertible Neural Networks. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, pages 1792–1800. PMLR.

Belkin, M., Rakhlin, A., and Tsybakov, A. B. (2019). Does data interpolation contradict statistical optimality? In *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, pages 1611–1619. PMLR.

Bernton, E., Jacob, P. E., Gerber, M., and Robert, C. P. (2019). Approximate Bayesian computation with the Wasserstein distance. *Journal of the Royal Statistical Society: Series B*, 81(2):235–269.

Bhadra, A., Datta, J., Polson, N., Sokolov, V., and Xu, J. (2021). Merging two cultures: Deep and statistical learning. *arXiv preprint arXiv:2110.11561*.

Bond-Taylor, S., Leach, A., Long, Y., and Willcocks, C. G. (2022). Deep Generative Modelling: A Comparative Review of VAEs, GANs, Normalizing Flows, Energy-Based and Autoregressive Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(11):7327–7347.

Brenier, Y. (1991). Polar factorization and monotone rearrangement of vector-valued functions. *Communications on Pure and Applied Mathematics*, 44(4):375–417.

Brillinger, D. R. (2012). A Generalized Linear Model With "Gaussian" Regressor Variables. In Guttorp, P. and Brillinger, D., editors, *Selected Works of David Brillinger*, Selected Works in Probability and Statistics, pages 589–606. Springer, New York, NY.

Carlier, G., Chernozhukov, V., and Galichon, A. (2016). Vector quantile regression: An optimal transport approach. *The Annals of Statistics*, 44(3):1165–1192.

Chernozhukov, V., Chetverikov, D., Demirer, M., Duflo, E., Hansen, C., Newey, W., and Robins, J. (2018). Double/debiased machine learning for treatment and structural parameters. *The Econometrics Journal*, 21(1):C1–C68.

Chernozhukov, V. and Hansen, C. (2005). An IV model of quantile treatment effects. *Econometrica*, 73(1):245–261.

Coppejans, M. (2004). On Kolmogorov's representation of functions of several variables by functions of one variable. *Journal of Econometrics*, 123(1):1–31.

Cox, D. R. (1972). Regression models and life-tables. *Journal of the Royal Statistical Society: Series B (Methodological)*, 34(2):187–202.

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

Dabney, W., Ostrovski, G., Silver, D., and Munos, R. (2018). Implicit Quantile Networks for Distributional Reinforcement Learning.

Dabney, W., Rowland, M., Bellemare, M. G., and Munos, R. (2017). Distributional Reinforcement Learning with Quantile Regression.

DiCiccio, T. J. and Efron, B. (1996). Bootstrap confidence intervals. *Statistical science*, 11(3):189–228.

Diggle, P. J. and Gratton, R. J. (1984). Monte Carlo Methods of Inference for Implicit Statistical Models. *Journal of the Royal Statistical Society. Series B (Methodological)*, 46(2):193–227.

Dinh, L., Krueger, D., and Bengio, Y. (2015). NICE: Non-linear Independent Components Estimation.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. (2017). Density estimation using Real NVP.

Donahue, J., Krähenbühl, P., and Darrell, T. (2017). Adversarial feature learning. In *International Conference on Learning Representations (ICLR)*.

Drovandi, C. C., Pettitt, A. N., and Faddy, M. J. (2011). Approximate Bayesian computation using indirect inference. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 60(3):317–337.

Drovandi, C. C., Pettitt, A. N., and Lee, A. (2015). Bayesian Indirect Inference Using a Parametric Auxiliary Model. *Statistical Science*, 30(1):72–95.

Dumoulin, V., Belghazi, I., Poole, B., Mastropietro, O., Lamb, A., Arjovsky, M., and Courville, A. (2017). Adversarially learned inference. In *International Conference on Learning Representations (ICLR)*.

Durkan, C., Murray, I., and Papamakarios, G. (2020). On contrastive learning for likelihood-free inference. In *International Conference on Machine Learning*, pages 2771–2781.

Efron, B. (1982). *The jackknife, the bootstrap and other resampling plans*. SIAM.

Efron, B. (1992). Bootstrap methods: another look at the jackknife. In *Breakthroughs in statistics: Methodology and distribution*, pages 569–593. Springer.

Efron, B. and Tibshirani, R. J. (1994). *An introduction to the bootstrap*. Chapman and Hall/CRC.

Fadikar, Arindam., Higdon, Dave., Chen, Jiangzhuo., Lewis, Bryan., Venkatramanan, Srinivasan., and Marathe, Madhav. (2018). Calibrating a Stochastic, Agent-Based Model Using Quantile-Based Emulation. *SIAM/ASA Journal on Uncertainty Quantification*, 6(4):1685–1706.

Geffner, T., Papamakarios, G., and Mnih, A. (2023). Compositional score modeling for simulation-based inference. *International Conference on Machine Learning*, pages 11098–11116.

Gelfand, A. E. (2000). Gibbs sampling. *Journal of the American statistical Association*, 95(452):1300–1304.

Gneiting, T. and Raftery, A. E. (2007). Strictly proper scoring rules, prediction, and estimation. *Journal of the American statistical Association*, 102(477):359–378.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2020). Generative adversarial networks. *Communications of the ACM*, 63(11):139–144.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. *Advances in neural information processing systems*, 27.

Greenberg, D., Nonnenmacher, M., and Macke, J. (2019). Automatic posterior transformation for likelihood-free inference. In *International Conference on Machine Learning*, pages 2404–2414.

Hahn, P. R., Murray, J. S., and Carvalho, C. M. (2020). Bayesian Regression Tree Models for Causal Inference: Regularization, Confounding, and Heterogeneous Effects (with Discussion). *Bayesian Analysis*, 15(3):965–1056.

Hannig, J., Iyer, H., Lai, R. C., and Lee, T. C. (2016). Generalized fiducial inference: A review and new results. *Journal of the American Statistical Association*, 111(515):1346–1361.

Hermans, J., Begy, V., and Louppe, G. (2020). Likelihood-free MCMC with amortized approximate ratio estimators. In *International Conference on Machine Learning*, pages 4239–4248.

Higdon, D., Gattiker, J., Williams, B., and Rightley, M. (2008). Computer model calibration using high-dimensional output. *Journal of the American Statistical Association*, 103(482):570–583.

Ho, J., Jain, A., and Abbeel, P. (2020). Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851.

Ho, J. and Salimans, T. (2022). Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*.

Hyvärinen, A. (2005). Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6:695–709.

Igelnik, B. and Parikh, N. (2003). Kolmogorov's spline network. *IEEE Transactions on Neural Networks*, 14(4):725–733.

Jiang, B., Wu, T.-Y., Zheng, C., and Wong, W. H. (2017). Learning Summary Statistic For Approximate Bayesian Computation Via Deep Neural Network. *Statistica Sinica*, 27(4):1595–1618.

Jiang, B., Wu, Tung-Yu, and Wing Hung Wong (2018). Approximate Bayesian Computation with Kullback-Leibler Divergence as Data Discrepancy. In *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, pages 1711–1721. PMLR.

Kallenberg, O. (1997). *Foundations of Modern Probability*. Springer, 2nd ed. edition edition.

Kennedy, M. C. and O'Hagan, A. (2001). Bayesian calibration of computer models. *Journal of the Royal Statistical Society: Series B*, 63(3):425–464.

Kingma, D. P. and Welling, M. (2014). Stochastic gradient vb and the variational auto-encoder. In *Second international conference on learning representations, ICLR*, volume 19, page 121.

Kingma, D. P. and Welling, M. (2022). Auto-Encoding Variational Bayes.

Kingma, D. P., Welling, M., et al. (2019). An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392.

Koenker, R. and Bassett Jr, G. (1978). Regression quantiles. *Econometrica*, 46(1):33–50.

Kolmogorov, A. N. (1957). On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition. *Doklady Akademii Nauk SSSR*, 114:953–956.

Kolmogorov, AN. (1942). Definition of center of dispersion and measure of accuracy from a finite number of observations (in Russian). *Izv. Akad. Nauk SSSR Ser. Mat.*, 6:3–32.

Kruse, J., Detommaso, G., Köthe, U., and Scheichl, R. (2021). HINT: Hierarchical Invertible Neural Transport for Density Estimation and Bayesian Inference. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(9):8191–8199.

Larsen, A. B. L., Sønderby, S. K., Larochelle, H., and Winther, O. (2016). Autoencoding beyond pixels using a learned similarity metric. In *International Conference on Machine Learning*, pages 1558–1566. PMLR.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444.

Lei, J., G'Sell, M., Rinaldo, A., Tibshirani, R. J., and Wasserman, L. (2018). Distribution-free predictive inference for regression. *Journal of the American Statistical Association*, 113(523):1094–1111.

Lueckmann, J.-M., Boelts, J., Greenberg, D., Goncalves, P., and Macke, J. (2021). Benchmarking simulation-based inference. In *Proceedings of the 24th International Conference on Artificial Intelligence and Statistics*, pages 343–351.

Lueckmann, J.-M., Goncalves, P. J., Bassetto, G., Öcal, K., Nonnenmacher, M., and Macke, J. H. (2017). Flexible statistical inference for mechanistic models of neural dynamics. *Advances in Neural Information Processing Systems*, 30.

MacKay, D. J. (1999). Maximum likelihood and covariant algorithms for independent component analysis.

Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., and Frey, B. (2015). Adversarial autoencoders. *arXiv preprint arXiv:1511.05644*.

Marin, J.-M., Pudlo, P., Robert, C. P., and Ryder, R. J. (2012). Approximate Bayesian computational methods. *Statistics and Computing*, 22(6):1167–1180.

McCann, R. J. (1995). Existence and uniqueness of monotone measure-preserving maps. *Duke Mathematical Journal*, 80(2):309–323.

Mirza, M. and Osindero, S. (2014). Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*.

Montanelli, H. and Yang, H. (2020). Error bounds for deep ReLU networks using the Kolmogorov–Arnold superposition theorem.

Moon, S. J., Jeon, J.-J., Lee, J. S. H., and Kim, Y. (2021). Learning multiple quantiles with neural networks. *Journal of Computational and Graphical Statistics*, 30(4):1238–1248.

Müller, T., Mcwilliams, B., Rousselle, F., Gross, M., and Novák, J. (2019). Neural Importance Sampling. *ACM Trans. Graph.*, 38(5):145:1–145:19.

Nunes, M. A. and Balding, D. J. (2010). On Optimal Selection of Summary Statistics for Approximate Bayesian Computation. *Statistical Applications in Genetics and Molecular Biology*, 9(1).

Ostrovski, G., Dabney, W., and Munos, R. (2018). Autoregressive Quantile Networks for Generative Modeling.

Padilla, O. H. M., Tansey, W., and Chen, Y. (2022). Quantile regression with ReLU networks: Estimators and minimax rates. *The Journal of Machine Learning Research*, 23(1):247:11251–247:11292.

Papamakarios, G. and Murray, I. (2016). Fast \epsilon -free Inference of Simulation Models with Bayesian Conditional Density Estimation. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

Papamakarios, G., Sterratt, D., and Murray, I. (2019). Sequential neural likelihood: Fast likelihood-free inference with autoregressive flows. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 837–848. PMLR.

Park, M., Jitkrittum, W., and Sejdinovic, D. (2016). K2-ABC: Approximate Bayesian Computation with Kernel Embeddings. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, pages 398–407. PMLR.

Parzen, E. (2004). Quantile Probability and Statistical Data Modeling. *Statistical Science*, 19(4):652–662.

Pastorello, S., Patilea, V., and Renault, E. (2003). Iterative and recursive estimation in structural nonadaptive models. *Journal of Business & Economic Statistics*, 21(4):449–509.

Polson, N., Sokolov, V., and Xu, J. (2021). Deep Learning Partial Least Squares. *arXiv preprint arXiv:2106.14085*.

Polson, N. G. and Ročková, V. (2018). Posterior Concentration for Sparse Deep Learning. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Radev, S. T., Mertens, U. K., Voss, A., Ardizzone, L., and Köthe, U. (2020). BayesFlow: Learning complex stochastic models with invertible neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 33(4):1452–1466.

Rezende, D. and Mohamed, S. (2015a). Variational Inference with Normalizing Flows. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 1530–1538. PMLR.

Rezende, D. J. and Mohamed, S. (2015b). Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*.

Romano, Y., Patterson, E., and Candès, E. (2019). Conformalized quantile regression. In *Advances in Neural Information Processing Systems*, volume 32.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. (2016). Improved techniques for training GANs. *Advances in Neural Information Processing Systems*, 29.

Schmidt-Hieber, J. (2020). Nonparametric regression using deep neural networks with ReLU activation function. *The Annals of Statistics*, 48(4):1875–1897.

Schultz, L., Auld, J., and Sokolov, V. (2022). Bayesian Calibration for Activity Based Models. *arXiv preprint arXiv:2203.04414*.

Sharrock, L., Simons, J., Liu, S., and Sherlock, C. (2024). Sequential neural score estimation: Likelihood-free inference with conditional score based diffusion models. *International Conference on Machine Learning*.

Shen, G., Jiao, Y., Lin, Y., Horowitz, J. L., and Huang, J. (2021). Deep Quantile Regression: Mitigating the Curse of Dimensionality Through Composition.

Shen, X. and Meinshausen, N. (2024). Engression: Extrapolation through the lens of distributional regression. *Journal of the Royal Statistical Society: Series B*.

Sisson, S. A., Fan, Y., and Beaumont, M. A. (2018). Overview of abc. In *Handbook of approximate Bayesian computation*, pages 3–54. Chapman and Hall/CRC.

Smith, B. J. (2007). boa: an r package for mcmc output convergence assessment and posterior inference. *Journal of statistical software*, 21:1–37.

Sohl-Dickstein, J., Weiss, E. A., Maheswaranathan, N., and Ganguli, S. (2015). Deep Unsupervised Learning using Nonequilibrium Thermodynamics.

Song, Y., Dhariwal, P., Chen, M., and Sutskever, I. (2023). Consistency models. In *International Conference on Machine Learning*, pages 32211–32252.

Song, Y., Meng, C., and Ermon, S. (2019). MintNet: Building Invertible Neural Networks with Masked Convolutions. *arXiv:1907.07945 [cs, stat]*.

Song, Y., Sohl-Dickstein, J., Kingma, D. P., Kumar, A., Ermon, S., and Poole, B. (2021). Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*.

Stroud, J. R., Müller, P., and Polson, N. G. (2003). Nonlinear state-space models with state-dependent variances. *Journal of the American Statistical Association*, 98(462):377–386.

Székely, G. J. and Rizzo, M. L. (2013). Energy statistics: A class of statistics based on distances. *Journal of Statistical Planning and Inference*, 143(8):1249–1272.

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

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

Trippe, B. L. and Turner, R. E. (2018). Conditional Density Estimation with Bayesian Normalising Flows. *arXiv:1802.04908 [stat]*.

Villani, C. (2009). *Optimal Transport: Old and New*, volume 338. Springer.

Vovk, V., Gammerman, A., and Shafer, G. (2005). *Algorithmic Learning in a Random World*. Springer.

Xie, M.-g. and Singh, K. (2013). Confidence distribution, the frequentist distribution estimator of a parameter: A review. *International Statistical Review*, 81(1):3–39.

Yoon, J., Jordon, J., and van der Schaar, M. (2018). GANITE: Estimation of individualized treatment effects using generative adversarial nets. In *International Conference on Learning Representations*.

A Bayes Risk

The bounds in this appendix apply to the parameter-generation class where the unknown function f is the posterior map and θ is the target. This appendix uses the standard nonparametric regression convention: X_i denotes the input (corresponding to the data summary $S(Y_i)$ in the main text) and Y_i denotes the response (corresponding to the parameter θ_i in the main text); these differ from the convention of the main text where Y_i represents observed data and θ_i is the parameter. Consider the nonparametric regression model $Y_i = f(X_i) + \varepsilon_i$, where $X_i = (x_{1i}, \ldots, x_{di}) \in [0,1]^d$, and the goal is to estimate the unknown multivariate function $f : [0,1]^d \to \mathbb{R}$. A natural measure of estimation accuracy is the integrated squared error, defined as

$$R(f, \hat{f}_N) = \mathbb{E}_{X,Y}\left[\|f - \hat{f}_N\|^2\right],$$

where $\|\cdot\|$ denotes the $L^2(P_X)$-norm and $\hat{f}_N$ is an estimator based on a sample of size N.

A common strategy is to construct $\hat{f}_N$ as a regularized solution to the empirical risk minimization problem:

$$\hat{f}_N = \arg\min_{f \in \mathcal{F}} \left\{ \frac{1}{N} \sum_{i=1}^{N} (Y_i - f(X_i))^2 + \lambda\, \phi(f) \right\},$$

where $\lambda > 0$ is a regularization parameter and $\phi(f)$ is a penalty functional controlling model complexity. Such estimators often correspond to the mode of a posterior distri-

bution under a Bayesian prior, resulting in a regularized maximum a posteriori estimate. Under appropriate smoothness and complexity conditions, these estimators have a posterior concentration: the risk $R(f, \hat{f}_N)$ converges to zero at a rate determined by the regularity of f and the structure of the function class $\mathcal{F}$.

The posterior distribution $p(f \mid X_i, Y_i)$ can concentrate around the true function at near-optimal rates (up to a $\log N$ factor). For instance, Barron (1993) shows that Fourier-based models can achieve convergence rates of order $O(N^{-1/2})$ in mean squared error. More generally, for functions f belonging to a β-Hölder class, the minimax risk over all estimators satisfies

$$\inf_{\hat{f}} \sup_{f} R(f, \hat{f}_N) = O_p\left(N^{-2\beta/(2\beta+d)}\right),$$

where d is the input dimension and $\hat{f}_N$ denotes any estimator based on N observations. These rates reflect the curse of dimensionality: estimation becomes harder as d increases. By restricting the class of functions, better rates can be obtained; when the function class has low-dimensional compositional structure, the effective dimension d in the rate can be much smaller than the ambient dimension, mitigating the curse of dimensionality. For example, it is common to consider the class of linear superpositions, also called ridge functions, or projection pursuit.

Consider the nonparametric regression model $Y_i = f(X_i) + \varepsilon_i$, where f is an unknown function to be estimated from data $(X_i, Y_i)_{i=1}^N$. Suppose that f is a composite function,

$$f = g_L \circ g_{L-1} \circ \cdots \circ g_0,$$

where each component g_ℓ is a Hölder-smooth function of d_ℓ variables with smoothness index β_ℓ. That is, for all $x, y \in \mathbb{R}^{d_\ell}$,

$$|g_\ell(x) - g_\ell(y)| \leq C\|x - y\|^{\beta_\ell}.$$

Under this assumption, recent results show that deep neural network estimators can achieve the convergence rate

$$O\left(\max_{1\leq \ell \leq L} N^{-2\beta_\ell^*/(2\beta_\ell^*+d_\ell)}\right), \quad \text{where} \quad \beta_\ell^* = \beta_\ell \prod_{k=\ell+1}^{L} \min(\beta_k, 1).$$

This bound shows how the local smoothness and dimensionality of each layer interact to determine the overall estimation rate. As an example, consider a generalized additive

model of the form

$$f_0(x) = h\left(\sum_{p=1}^{d} f_{0p}(x_p)\right),$$

where h is a Lipschitz function and each f_{0p} is a univariate smooth function. This structure can be viewed as a composition of three low-dimensional functions: $g_0(z) = h(z)$ with $z \in \mathbb{R}$; $g_1(x_1, \ldots, x_d) = (f_{01}(x_1), \ldots, f_{0d}(x_d))$; and $g_2(y_1, \ldots, y_d) = \sum_{p=1}^{d} y_p$. Since each component operates over one-dimensional or additive functions and h is Lipschitz, the resulting estimator achieves the rate $O(N^{-1/3})$, which is independent of the dimension d. Schmidt-Hieber (2020) show that deep rectified-linear-unit networks achieve the optimal rate $O(N^{-1/3})$ under similar assumptions. Coppejans (2004) finds a convergence rate of $O(N^{-3/7})$, equivalently $O(N^{-3/(2\beta+d)})$ with $\beta = 3$ and $d = 1$, for three-times differentiable cubic B-splines. Igelnik and Parikh (2003) derives a parametric rate $O(N^{-1})$ for Kolmogorov spline networks.

B Bayes Rule for Quantiles

The compositional quantile identity developed in this appendix underlies the generative Bayesian computation framework for parameter generation (Section 5) and the forward-emulator quantile network for predictive inference (Section 6); the same left-continuous generalized inverse (Parzen, 2004) appears as $G(q, \tilde{s}, a)$ in the counterfactual generator for outcome generation in Section 2. Parzen (2004) shows that quantile methods are direct alternatives to density estimation. Following the convention of this work, we use u for the quantile level in this appendix (corresponding to τ in the main text). Given the conditional cumulative distribution function $F_{\theta|y}$, assumed to be continuous and non-decreasing, the associated quantile function is defined as

$$Q_{\theta|y}(u) := F^{-1}_{\theta|y}(u) = \inf\left\{\theta : F_{\theta|y}(\theta) \geq u\right\},$$

which is left-continuous and non-decreasing. A fundamental property established by Parzen (2004) is the probability integral transform

$$\theta \overset{d}{=} Q_{\theta|y}(U), \qquad U \sim \text{Unif}(0,1),$$

which shows that the quantile function generates draws from the target distribution by transforming uniform random variables: monotonicity of the quantile map aligns the order statistics of the parameter with those of the baseline distribution, so sampling reduces

to evaluating $Q_{\theta|y}$ at uniform draws. Consider now a non-decreasing, left-continuous function $g(y)$, with generalized inverse defined by

$$g^{-1}(z) = \sup\{y : g(y) \leq z\}.$$

Then the quantile of a transformed variable satisfies the composition identity

$$Q_{g(Y)}(u) = g(Q_Y(u)),$$

illustrating that quantile operations are compositional.

Proof. Let g be strictly increasing and continuous. Then:

$$F_{g(Y)}(z) = p(g(Y) \leq z) = p(Y \leq g^{-1}(z)) = F_Y(g^{-1}(z)).$$

The quantile function of $g(Y)$ is:

$$Q_{g(Y)}(u) = \inf\left\{z : F_{g(Y)}(z) \geq u\right\} = \inf\left\{z : F_Y(g^{-1}(z)) \geq u\right\}.$$

Substitute $z = g(y)$, which implies $y = g^{-1}(z)$:

$$Q_{g(Y)}(u) = \inf\{g(y) : F_Y(y) \geq u\} = g\left(\inf\{y : F_Y(y) \geq u\}\right) = g(Q_Y(u)).$$

□

This property supports the interpretation of nested quantile transformations as layered representations, akin to deep learning architectures. In Bayesian models, this compositional structure underlies the conditional quantile representation of the posterior:

$$Q_{\theta|Y=y}(u) = Q_\theta(s), \quad \text{where} \quad s = Q_{F(\theta)|Y=y}(u).$$

To determine s, observe that

$$u = p(F(\theta) \leq s \mid Y = y) = p(\theta \leq Q_\theta(s) \mid Y = y) = F_{\theta|Y=y}(Q_\theta(s)),$$

which confirms that $s = F^{-1}_{F(\theta)|Y=y}(u)$. This recursive identity provides a quantile-based method for posterior updating, bypassing the need for explicit density evaluation.

C Comparative Analysis

The three generator classes share a common representation theorem (noise outsourcing theorem of Kallenberg (1997) applied to different variable pairs), but condition on different objects, rely on different identification assumptions, and answer different statistical questions. Counterfactual outcome generators condition on covariates and treatment states and ask what outcome distribution would arise under a specified intervention. Parameter–outcome generators condition on observed outcomes and ask what parameter values are plausible under a forward model. Predictive generators condition on observed covariates, design points, or simulator inputs and ask what future outcomes are plausible. Methods can move across classes only when their loss function, conditioning variables, and output interpretation are changed accordingly. For example, a conditional flow can be a posterior estimator in the second class or a predictive density estimator in the third; a diffusion model can be an outcome generator in the first class or a conditional posterior sampler in the second; a quantile network can estimate counterfactual outcome quantiles, posterior quantiles, or predictive quantiles depending on whether it is trained on $(\tilde{S}(X), A, Y)$, (Y, θ), or (X, Y), respectively. Table 2 records the respective methods, training object, main output, and main limitations for each class.

Generator class	Training object	Methods	Main statistical output	Main limitation
Counterfactual outcome distributions	Observed or simulated triples $(\tilde{S}(X), A, Y)$; more generally (c, Y)	Conditional adversarial generators, diffusion models, conditional variational autoencoders, conditional flows, quantile generators	Conditional outcome distribution, potential-outcome distribution, treatment-effect functionals	Identification depends on causal assumptions; calibration and support overlap must be checked
Parameter–outcome generators	Simulated pairs (θ, Y) from a prior and forward simulator	Approximate Bayesian computation, fiducial inference, conditional flows, conditional diffusion models, generative Bayesian computation	Posterior draws, posterior quantiles, or posterior density approximation	Summary choice, invertibility, sampling cost, or absence of closed-form density depending on method
Predictive generators	Input–output pairs (X, Y) from data or simulator runs	Gaussian process emulators, quantile networks, conditional flows, conditional variational autoencoders, conditional adversarial and diffusion generators	Predictive distribution, prediction intervals, conditional quantiles	Covariate shift, conditional calibration, and computational scaling

Table 2: Three generator classes used throughout the paper. The same architecture may appear in more than one row, but its statistical meaning changes with the training object and target distribution.

Table 3 additionally summarizes class membership for the eight architectures reviewed

above. A primary assignment indicates that the architecture is naturally suited to that class; a secondary assignment indicates that the architecture can be adapted to that class with a change of training object, loss function, or conditioning structure; the absence of an entry indicates that the architecture does not naturally serve that class.

Architecture	Counterfactual outcome	Parameter–outcome	Predictive
Approximate Bayesian computation	–	Primary	–
Fiducial inference	–	Primary	–
Variational autoencoder	Secondary	Secondary	Secondary
Normalizing flow	Secondary	Primary	Secondary
Generative adversarial network	Primary	–	Secondary
Diffusion model	Primary	Secondary	Secondary
Gaussian process emulator	–	Secondary	Primary
Generative Bayesian computation	Secondary	Primary	Secondary

Table 3: Membership of architectures across the three generator classes. Primary entries indicate that the architecture is naturally suited to the class; secondary entries indicate that the architecture serves the class after a change of training object, loss function, or conditioning structure. Empty entries indicate that the architecture does not naturally serve the class.

A third comparison records the computational properties of the individual architectures. Table 4 indexes each by primary class and by whether it is likelihood-free, provides exact density evaluation, amortizes inference across observations, handles multimodality, generates samples, requires an invertible architecture, or refines sequentially, together with approximate training and sampling costs. The central trade-off is between exact density evaluation (normalizing flows) and sample-only output (approximate Bayesian computation, generative adversarial networks, generative Bayesian computation), and between amortized estimators that evaluate at any new observation in one pass (variational autoencoders, normalizing flows, generative adversarial networks, generative Bayesian computation) and methods that iterate per observation (approximate Bayesian computation, diffusion models).

	ABC	VAE	NF/ICA	GAN	Diffusion	Fiducial	GBC
Primary class	2	1, 2, 3	1, 2, 3	1, 3	1, 2, 3	2	2
Likelihood-free	✓		✓†	✓	✓		✓
Exact density			✓				
Amortized inference		✓	✓	✓			✓
Handles multimodality	✓		∼‡	✓	✓	✓	✓
Generates samples	✓	✓	✓	✓	✓	✓	✓
Invertible architecture			Required				
Sequential refinement	✓				✓		
Training cost	Low	Medium	Medium	High	High	Low	Medium
Sampling cost	High	Low	Low	Low	High	High	Low

Table 4: Computational properties of the generative architectures, indexed by primary class: Cat 1 (counterfactual outcome), Cat 2 (parameter–outcome), Cat 3 (predictive). See Section 4 for a detailed summary of the algorithms. †Conditional normalizing flows can be trained on simulated pairs without likelihood evaluation, though the flow itself provides exact density computation. ‡Normalizing flows can represent mildly multimodal distributions but may struggle with well-separated modes because of the continuity of the bijective map.

The pattern of primary and secondary memberships in Table 3 reflects three architectural choices: invertibility, the form of the conditioning input, and the form of the output. Approximate Bayesian computation and fiducial inference are built on inverting a forward data-generating equation; they are tightly coupled to the parameter–outcome class because the simulated table $\{(\theta_i, Y_i)\}_{i=1}^N$ is their natural training object. They have no direct counterfactual or forward-predictive analogue. The two methods differ in interpretation rather than mechanism: approximate Bayesian computation treats the resulting distribution as the Bayesian posterior under a specified prior, while generalized fiducial inference (Hannig et al., 2016) treats it as a confidence distribution induced by the structural equation, with no prior input; in regular models the two coincide under Jeffreys' prior and otherwise differ by a Jacobian. Generative adversarial networks and diffusion models, in contrast, are designed to match an unconditional or conditional outcome distribution; they belong primarily to the first class. Their use in the second class requires explicit conditioning on observed data and a posterior target, which has been pursued in conditional diffusion methods for simulation-based inference (Geffner et al., 2023; Sharrock et al., 2024); this is a secondary use because it changes the training object and loss. Gaussian process emulators are primarily predictive: they fit a forward map from input parameters to outcomes; their use as posterior estimators requires an additional MCMC

calibration step (the quantile kriging construction of Fadikar et al. 2018 used in Section 6), which is a secondary use.

Normalizing flows and generative Bayesian computation move most easily across classes. A normalizing flow is invertible, so it can serve as a conditional density estimator for both the outcome distribution and the posterior distribution. Generative Bayesian computation is not invertible but trains a quantile function; by exchanging the roles of θ and Y (Section 5), the same architecture serves either as a posterior generator or a predictive generator, and by changing the conditioning to $(\tilde{S}(X), A, Y)$ it serves as a counterfactual outcome generator. Variational autoencoders straddle all three classes through the encoder–decoder structure: the encoder approximates a conditional distribution that can be a posterior, an outcome distribution, or a predictive distribution depending on what is encoded. Read against classical statistical structure, the variational autoencoder is a nonlinear factor-analysis model: the Gaussian latent prior is a modeling convention (not the Bayesian prior of the inferential problem), the decoder is a nonlinear analogue of the factor-loading map, and the evidence lower bound trades exact likelihood for tractability. The primary point is that the architecture is not the determinant of which class a method belongs to: the training object and conditioning structure are, and any architecture that can represent a conditional distribution can in principle serve any of the three classes provided the appropriate training object and loss are used.

D Computational Cost Comparison

Table 5 compares the computational requirements of four methods for the parameter-outcome generator class, illustrated with the Ebola application ($d = 5$ parameters, $N =$ 10,000 simulated trajectories from Fadikar et al. 2018). For brevity, we abbreviate generative Bayesian computation as GBC in the table and surrounding discussion.

Method	Upfront simulator calls	Fit / calibration cost (paid once)	Per new observation (after fit)	MCMC
GBC (this paper)	N	Neural network fit	B network evaluations for B posterior draws	No
ABC, nearest-neighbor	N	None	k-NN query over the N stored pairs	No
Quantile kriging (Fadikar 2018)	N	Gaussian process fit	Fresh MCMC chain of length N_{chain}	Yes
MCMC with simulated likelihood	0 upfront	None	$O(N_{\text{chain}})$ fresh simulator calls per chain	Yes

Table 5: Computational requirements for the parameter-outcome generator class, Ebola application ($d = 5$, $N = 10{,}000$). Columns separate the upfront simulator budget, the one-time training cost, and the marginal cost paid for each new observation y after fitting. GBC, ABC, and quantile kriging share the same upfront simulator budget; GBC and ABC amortize fitting over future observations. The critical difference is the per-observation cost after fit: GBC and ABC require no further simulator calls, while quantile kriging requires a fresh Markov chain per observation and likelihood-based MCMC requires $O(N_{\text{chain}})$ fresh simulator calls per observation.

The shared simulation budget N is the dominant upfront cost for GBC, ABC, and quantile kriging. After fitting, each GBC posterior draw at a new y requires one network forward evaluation, so producing $B = 1000$ draws costs B evaluations of a fixed-size network. For quantile kriging (Fadikar et al., 2018), posterior inference for a new y requires a fresh Markov chain run against the fitted Gaussian process as a likelihood surrogate. The per-observation cost is therefore $O(N_{\text{chain}})$ steps. For standard MCMC that calls the simulator directly at each iteration, the per-observation cost is $O(N_{\text{chain}} \times n_{\text{chains}})$ simulator evaluations; in the Ebola setting, where each ABM run costs several seconds, this makes per-observation MCMC infeasible at scale. Posterior uncertainty is quantified in GBC through the trained quantile network: the marginal per-coordinate 90% credible interval for parameter θ_j at observation y is $[\hat{G}(0.05, y), \hat{G}(0.95, y)]$. The empirical marginal per-coordinate coverage reported in Section 6.1 (88–90% for nominal 90%) is comparable to coverage reported for quantile kriging by Fadikar et al. (2018), with the advantage that GBC requires no covariance function specification and no per-observation Markov chain.

E Quantile Transport versus Adversarial Generators

This appendix develops the contrast between the generative Bayesian computation quantile-transport construction of Section 5 and the adversarial encoder–decoder family. Generative Bayesian computation and the adversarial encoder–decoder both define a generative map that pushes a simple baseline measure forward onto the target conditional law, but they differ in what map is learned and what objective is optimized, and the differences favor the quantile-transport construction for inference. The quantile network of Section 5 learns a *deterministic* transport map, the conditional quantile (or, for vector parameters, the Brenier) map, by minimizing a fixed proper scoring rule. A generative adversarial network instead learns an implicit generator through a two-player game. Recall the adversarial construction of Section 4: a generator G_{θ_g} driven by an abstract latent prior $Z \sim p(Z)$ and a discriminator D_{θ_d}, trained at the two-player saddle point

$$\min_{\theta_g} \max_{\theta_d} \; \mathbb{E}_{Y \sim p_{\text{obs}}}\big[\log D_{\theta_d}(Y)\big] + \mathbb{E}_{Z \sim p_Z}\big[\log\big(1 - D_{\theta_d}(G_{\theta_g}(Z))\big)\big].$$

For a fixed generator the optimal discriminator is $D^\star(Y) = p_{\text{obs}}(Y)/(p_{\text{obs}}(Y) + p_G(Y))$, and substituting it shows the generator minimizes a divergence,

$$V(G_{\theta_g}, D^\star) = 2\,\mathrm{JS}\big(p_{\text{obs}} \,\|\, p_G\big) - \log 4,$$

while the Wasserstein variant (Arjovsky et al., 2017) replaces the Jensen–Shannon divergence with the Kantorovich–Rubinstein dual of $W_1(p_{\text{obs}}, p_G)$. In both cases the quantity actually descended is an *adversarial estimate* of a divergence: it depends on an inner maximization over θ_d that a finite-capacity discriminator, trained for finitely many steps, only approximates. Encoder–decoder variants, bidirectional adversarial inference (Donahue et al., 2017; Dumoulin et al., 2017), the variational-autoencoder–adversarial hybrid (Larsen et al., 2016), and adversarial autoencoders (Makhzani et al., 2015), add an encoder and sometimes a reconstruction term, but the latent code remains unidentified: many distinct generator–encoder pairs induce the same pushforward p_G.

Generative Bayesian computation solves $\min_w L(w)$ for a fixed risk L that is an expectation of a proper scoring rule and is convex in the network outputs $\hat{Q}(\tau, y)$, with a unique population minimizer equal to the true quantile map. The adversarial problem $\min_{\theta_g} \max_{\theta_d}$ has no single objective: its solution is a saddle point, and simultaneous or alternating gradient play need not converge even in convex–concave idealizations. This is the structural source of the mode collapse, oscillation, and hyperparameter sensitivity seen in adversarial training. The quantile-network gradient is the subgradient $\mathbf{1}\{\theta <$

$q\} - \tau \in [-\tau, 1-\tau]$, bounded and almost surely non-vanishing, so the objective always supplies a descent signal. The adversarial generator gradient $\nabla_{\theta_g} \log\bigl(1 - D_{\theta_d}(G_{\theta_g}(z))\bigr)$ vanishes whenever the discriminator saturates, precisely when p_G and p_{obs} have nearly disjoint support and the Jensen–Shannon divergence is locally constant at $\log 2$. Generative Bayesian computation has no regime in which the loss stops providing a gradient.

Additionally, the check loss and the energy score are strictly proper, so the quantile network is an ordinary M-estimator: as the simulation budget N grows and capacity increases, the fitted map converges to the true posterior quantile map under the empirical-risk-minimization rates established in Section 5. Consistency of adversarial networks instead requires the discriminator to attain its supremum at every generator update. With a finite, imperfectly trained discriminator the generator descends a biased surrogate of the divergence, and no analogous guarantee holds. This is the same gap flagged for the adversarial objective in the convergence-rate discussion of Section 5.

In generative Bayesian computation the latent is the uniform index U, the level τ *is* the probability level, and the target map is pinned down, by monotonicity in one dimension and by the convex Brenier potential in several, so the map is unique. In an adversarial model the generator is identified only up to its pushforward p_G; the encoder–decoder pair admits a large equivalence class of solutions, and the latent space carries no probabilistic ordering. This identifiability is what lets the quantile network report calibrated quantiles and credible intervals, which the adversarial latent does not support.

Because the fitted quantile map is monotone it is invertible, so generative Bayesian computation returns the conditional cumulative distribution function directly and the density through a scalar Jacobian, alongside samples, quantiles, and credible intervals from one trained object. A generative adversarial network is a pure sampler: it provides no tractable likelihood, and density or interval estimates require additional steps applied to its samples.

Generative Bayesian computation treats the data as a network input, so $\hat{Q}(\tau, y)$ returns the posterior for a new y in a single forward pass, with one network and one loss. A conditional adversarial model must inject y into both the generator and the discriminator, doubling the surface on which the two-player game can fail, and must balance generator and discriminator capacity and learning rates throughout training.

Aspect	GBC quantile–transport	GAN encoder–decoder
Object learned	quantile or Brenier transport map	implicit generator, optionally with an encoder
Optimization	$\min_w L(w)$, fixed convex risk	$\min_{\theta_g} \max_{\theta_d}$ saddle point
Objective	proper scoring rule (check or energy)	adversarial estimate of a divergence
Generator gradient	bounded, almost surely non-vanishing	vanishes when the discriminator saturates
Consistency	M-estimation guarantees	needs exact inner maximization
Latent	uniform U; level τ identified	abstract z, unidentified
Density and CDF	available (scalar Jacobian)	not tractable
Conditioning	native input, one forward pass	injected into both G and D
Networks to train	one	two, or three with an encoder
Failure modes	quantile crossing (preventable)	mode collapse, non-convergence

Table 6: Generative Bayesian computation (quantile transport) versus the adversarial encoder–decoder as estimators of a conditional distribution. The contrasts are developed in the text of this appendix; the adversarial construction is given in Section 4.

The well-defined one-dimensional quantile characterization relies on the total ordering of $\mathbb{R}$. In genuinely multivariate problems the relevant object is the Brenier map, monotone in the optimal-transport sense rather than coordinate-wise, and the practical objective becomes a multivariate proper scoring rule such as the energy score introduced in Section 5 rather than the scalar check loss; the structural advantages, deterministic transport, a fixed proper objective, no discriminator, and an identified map, carry over, but the idea of directly modeling the quantile function should be read in its transport-map generalization. These guarantees are also relative to the simulator: bias in the assumed prior or likelihood propagates into the fitted map, as it does in any simulation-based inference scheme.

F Counterfactual outcome distributions: a synthetic example

This subsection first develops the counterfactual outcome generator, its associated causal estimands, and the quantile treatment effects from the same formalism used elsewhere in the paper, and then illustrates the construction on a synthetic example with known ground truth so that the estimates of the conditional average treatment effects can be

tested against the truth. The two larger numerical comparisons of the paper, against existing methods on a real simulator, remain the inverse-inference application in Section 6.1 and the forward-emulation application in Section 6.2.

To illustrate the first generator class, consider a binary treatment program evaluation. A population of eligible individuals is assigned to a treatment arm ($A = 1$) or a control arm ($A = 0$). The outcome Y is a post-program labor-market measure such as annual earnings; the covariate vector X records baseline characteristics. The estimand of interest is the average treatment effect $\Delta = \mathbb{E}[Y(1) - Y(0)]$ and its conditional version $\Delta(x) = \mathbb{E}[Y(1) - Y(0) \mid X = x]$. The generator here is not a posterior generator over model parameters. It is the counterfactual outcome generator

$$G(q, \tilde{s}, a) = F^{-1}_{Y|\tilde{S}(X)=\tilde{s},\, A=a}(q), \qquad q \in (0,1), \quad a \in \{0,1\},$$

where $\tilde{S}(X)$ is a sufficient covariate reduction, for example the estimated propensity score or a low-dimensional learned summary. The training object is the observed causal triple

$$\{(\tilde{S}(X_i), A_i, Y_i)\}_{i=1}^{N}.$$

A quantile generator estimates G directly by the pinball loss. Conditional adversarial generators, conditional variational autoencoders, conditional diffusion models, and conditional flows target the same conditional outcome distribution with different numerical criteria.

Under consistency, unconfoundedness given $\tilde{S}(X)$, positivity, and the stable unit treatment value assumption, the conditional distribution of Y given $(\tilde{S}(X) = \tilde{s}, A = a)$ identifies the conditional distribution of $Y(a)$ given $\tilde{S}(X) = \tilde{s}$. Causal estimands are then functionals of the trained generator:

$$\Delta(\tilde{s}) = \int_0^1 \left[\widehat{G}(q, \tilde{s}, 1) - \widehat{G}(q, \tilde{s}, 0)\right] dq, \qquad \Delta = \int \Delta(\tilde{s})\, dF_{\tilde{S}(X)}(\tilde{s}).$$

Quantile treatment effects are recovered as $\Delta_q(\tilde{s}) = \widehat{G}(q, \tilde{s}, 1) - \widehat{G}(q, \tilde{s}, 0)$.

This setup illustrates the key distinction between the first and second generator classes. Bayesian approaches to program evaluation, such as Bayesian causal forest (Hahn et al., 2020), parameterize the response surface and recover treatment effects through MCMC posterior sampling. That places them in the second generator class: the target is a posterior over model parameters, and draws require a Markov chain whose length scales with the convergence properties of the sampler. The counterfactual outcome generator here

belongs to the first class: it targets the conditional outcome distribution directly without parameterizing the data-generating process, and each draw requires only one function evaluation after the generator has been fitted. Appendix D compares the computational cost of generator-based and MCMC-based methods in detail for the parameter-outcome generator task. The same pattern (MCMC requires a chain per observation, generator methods require one network evaluation) applies in the counterfactual outcome generator class.

To illustrate the performance of the generative Bayesian computation, we apply the quantile-network counterfactual generator to a one-dimensional synthetic data-generating process where the true treatment effect is known. The covariate $X \sim \mathcal{N}(0, 1)$, the treatment indicator $A \mid X \sim \text{Bernoulli}(\sigma(0.5\,X))$ with a smooth propensity (no extreme overlap), and the potential outcomes are $Y(0) = X + \varepsilon_0$ and $Y(1) = 1.5\,X + 1 + \varepsilon_1$ with $\varepsilon_0, \varepsilon_1 \sim \mathcal{N}(0, 0.5^2)$. The observed outcome is $Y = A\,Y(1) + (1 - A)\,Y(0)$. The true conditional average treatment effect is $\Delta(x) = 1 + 0.5\,x$ and the true average treatment effect is $\Delta = \mathbb{E}[\Delta(X)] = 1$. We fit an implicit quantile network with a cosine quantile embedding (Section 5) on $N = 2{,}000$ observed triples $\{(X_i, A_i, Y_i)\}$ by minimizing the pinball loss, then estimate the conditional treatment effect as $\hat{\Delta}(x) = \int_0^1 [\widehat{G}(q, x, 1) - \widehat{G}(q, x, 0)]\,dq$, approximated by an average over 200 quantile levels, and the average treatment effect as $\hat{\Delta}$, the mean of $\hat{\Delta}(\cdot)$ over a held-out covariate test set of size $n_{\text{test}} = 500$.

Across 20 Monte Carlo replicates, the estimator $\hat{\Delta}$ has mean 1.000 against the true value 1.000 (bias -0.0003, root mean squared error 0.033). Figure 12 shows the recovered conditional treatment-effect curve, which traces the true $\Delta(x) = 1 + 0.5\,x$ across the support of X, and the empirical distribution of $\hat{\Delta}$ around the true ATE. A 90% confidence interval on $\hat{\Delta}$ is constructed by a nonparametric bootstrap over the training data: for each replicate, the quantile network is retrained on $B = 20$ bootstrap resamples of $\{(X_i, A_i, Y_i)\}$ and $\hat{\Delta}^{(b)}$ is recomputed on the held-out test set. The 90% interval is the 5th to 95th percentile of the resulting bootstrap distribution. Empirical coverage of the true $\Delta = 1$ across the 20 replicates is 0.80 (16/20). With $B = 20$ bootstrap retrains the percentile interval is somewhat noisy, but the interval now reflects both Monte Carlo over the test covariates and the training-data sampling variability of the generator. The headline message of the illustration is the recovery of the conditional treatment-effect map together with the smallness of the average bias and root mean squared error: the counterfactual outcome generator, fit by a single pass of pinball-loss minimization, returns an estimator $\hat{\Delta}(x)$ that is essentially unbiased for the true $\Delta(x)$ on this data-generating process.

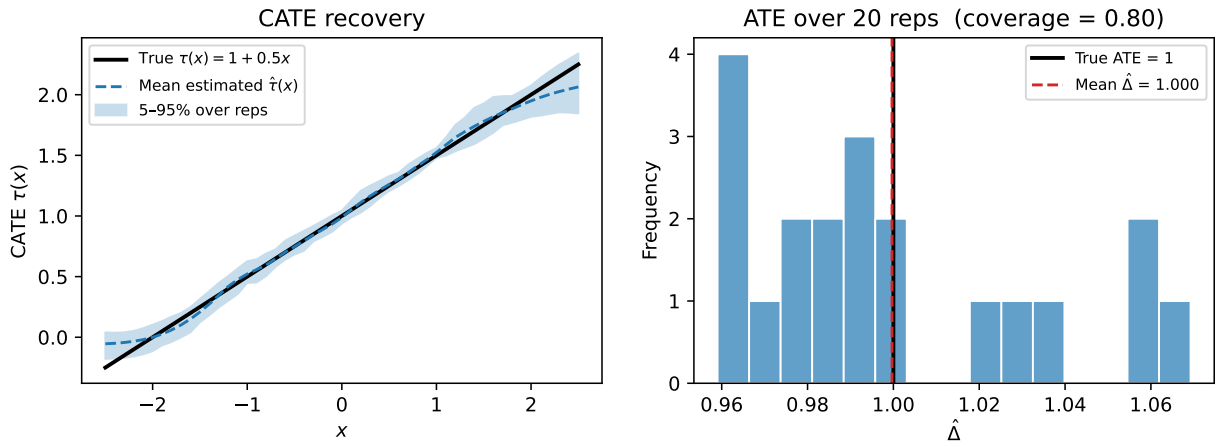

Figure 12: Counterfactual outcome generator on a synthetic one-dimensional example. *Left:* recovered conditional average treatment effect $\hat{\Delta}(x)$ across 20 replicates (mean as the dashed line; 5th–95th percentile band shaded) against the true $\Delta(x) = 1 + 0.5\,x$ (solid line). *Right:* empirical distribution of the estimated average treatment effect $\hat{\Delta}$ across 20 replicates; the true value $\Delta = 1$ is marked with a solid vertical line and the empirical mean 1.000 with a dashed line. Empirical 90% coverage of the training-data percentile bootstrap interval is 0.80 (16/20).

G Calibration Checks for the Ebola Inverse Problem

The Ebola coverage statistics of Section 6.1 are simulation-based calibration of the average posterior. To complement them, we report two diagnostics that test the approximation more directly: a closed-form conjugate benchmark where the exact posterior is available, and per-parameter simulation-based calibration rank histograms on the Ebola test set. We also report a comparison against a flow-based neural posterior estimator on the same simulated table.

Conjugate benchmark

Let $\theta \sim \text{Beta}(2, 2)$ and $Y_i \mid \theta \sim \text{Bernoulli}(\theta)$ for $i = 1, \ldots, 20$. The sufficient statistic $s = \sum_i Y_i$ is Bayes-sufficient and the exact posterior is $\theta \mid s \sim \text{Beta}(2 + s,\, 2 + 20 - s)$ in closed form. We train an implicit quantile network on $N = 50{,}000$ simulated (θ, s) pairs by minimizing the pinball loss, then draw 5000 posterior samples at each of $s_{\text{obs}} \in \{4, 10, 16\}$ and compare against the exact Beta density (Figure 13). Across the three values of s_{obs}, the Kolmogorov–Smirnov (KS) statistic between the generative Bayesian computation sample and the exact Beta cumulative distribution function is between 0.032 and 0.040; posterior means and standard deviations agree with the exact values to within 0.003. The exact

posterior is closely approximated by the generative Bayesian computation sample on this benchmark where the ground truth is available in closed form.

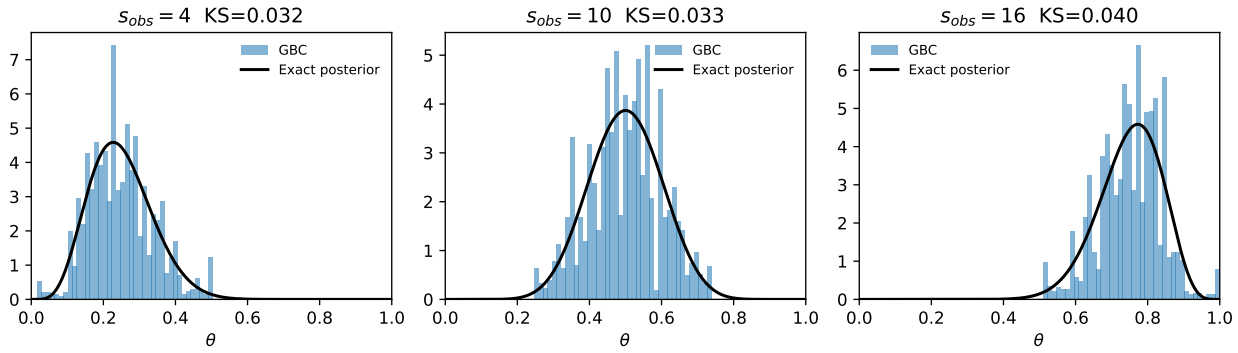

Figure 13: Conjugate Beta-Binomial benchmark with exact posterior available in closed form. Each panel overlays 5000 generative Bayesian computation posterior samples (blue histogram) against the exact Beta$(2 + s_{\text{obs}},\ 2 + 20 - s_{\text{obs}})$ density (black curve), at three values of the observed sufficient statistic. Kolmogorov–Smirnov statistics between the generative Bayesian computation sample and the exact posterior cumulative distribution function are 0.032–0.040.

Simulation-based calibration on the Ebola inverse problem

Talts et al. (2018) show that for a correctly calibrated posterior approximation, the ranks of the true parameter values among posterior draws across simulated (θ, Y) pairs are uniformly distributed. We retrain per-parameter quantile networks on 8000 of the 10,000 simulated Ebola trajectories using the same partial least squares summary, then for each of the 2000 held-out trajectories draw $B = 500$ posterior samples and compute the rank of the true θ_j among them. Figure 14 shows the resulting rank histograms with a 99% confidence band for uniformity. Table 7 reports a chi-square uniformity test with 19 degrees of freedom.

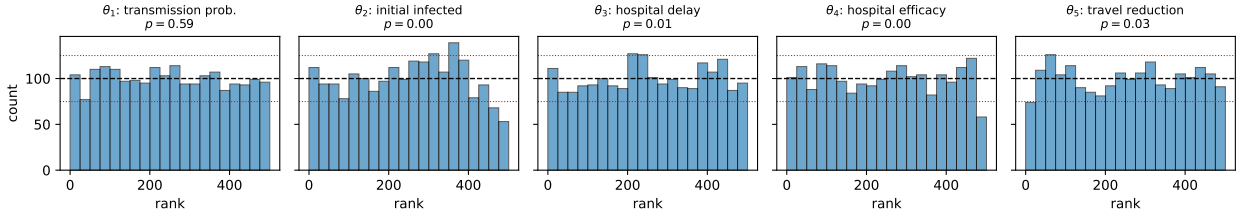

Figure 14: Simulation-based calibration rank histograms on the Ebola inverse problem (2000 held-out trajectories, $B = 500$ posterior draws per trial). Dashed line is the uniform expectation; red dotted lines are the 99% pointwise confidence band. Per-panel chi-square uniformity p-values are reported above each panel.

parameter	χ^2_{19}	p-value	mean rank
θ_1	17.0	0.59	247.8
θ_2	81.8	9×10^{-10}	245.9
θ_3	35.1	0.014	254.2
θ_4	41.6	0.002	246.6
θ_5	32.4	0.028	251.6

Table 7: Simulation-based calibration of the generative Bayesian computation posterior on the Ebola inverse problem, by parameter. Mean ranks are close to the uniform target of $B/2 = 250$ in every case, but the chi-square test rejects uniformity at the 0.05 level for θ_2, θ_3, θ_4, and (borderline) θ_5. This refines the coverage diagnostic of Section 6.1: marginal coverage averages over the test set can be near nominal even when the rank distribution is non-uniform.

The interpretation is: marginal coverage and simulation-based calibration are complementary, not equivalent. Mean rank near $B/2$ is consistent with coverage near nominal, but the rank distribution itself probes the shape of the posterior approximation, and simulation-based calibration detects structural mismatch in θ_2–θ_4 that the coverage statistic alone does not.

Neural posterior estimation baseline

The natural comparison is a flow-based neural posterior estimator on the same simulated table. We train an amortized neural posterior estimator with a Masked Autoregressive Flow (MAF) density estimator, using the SNPE-C implementation of Greenberg et al. (2019) from the `sbi` package (Tejero-Cantero et al., 2020) (single-round training on the full simulation budget, with no sequential refinement against the observed datum), the same 8000/2000 train/test split, and the same partial least squares summary as the quantile network; we then form 90% marginal credible intervals from 1000 posterior draws per test point. Table 8 reports the comparison.

method	θ_1	θ_2	θ_3	θ_4	θ_5
GBC (quantile network)	0.89	0.88	0.90	0.90	0.90
NPE flow (MAF, defaults)	0.82	0.84	0.71	0.79	0.74

Table 8: Empirical 90% marginal credible interval coverage on the Ebola inverse problem, by parameter. Both methods are trained on the same 8000 simulated (θ, Y) pairs with the same partial least squares summary. Coverage is evaluated on 2000 held-out trajectories. Quantile-based generative Bayesian computation matches or exceeds the masked autoregressive flow at default settings on every parameter. The comparison uses sbi v0.26.1 default hyperparameters for the flow; no claim is made that the flow cannot be improved with extensive tuning.

The quantile network has between 4 and 19 percentage points higher marginal coverage than the flow at default settings on every parameter. The largest gaps are on θ_3 (0.90 vs 0.71) and θ_5 (0.90 vs 0.74). The likely explanation is the one in Section 5: the pinball training criterion directly targets the conditional quantile function, which is the object that determines credible interval endpoints, while flow training targets the log-likelihood of the posterior and only indirectly controls quantile placement. This is a single comparison on one dataset and should be read as a reference point rather than a general statement about the two architectures.