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: May 17, 2025

Abstract

We review generative models that simulate large training datasets consisting of parameter–data pairs and use deep neural networks to learn mappings from observed outcomes to latent parameters. These methods frame posterior parameter prediction as a supervised learning problem and replace traditional sampling-based techniques, such as Markov Chain Monte Carlo. We emphasize the use of quantile-based and latent-variable architectures for estimating posterior parameter distributions, particularly in settings with high-dimensional data, nonlinear relationships, or intractable likelihoods. To demonstrate the practical relevance of these techniques, we apply a generative Bayesian computation framework to the well-known Ebola dataset. Our review highlights the flexibility, scalability, and efficiency of generative approaches in modern probabilistic modeling, and outlines directions for further research in likelihood-free inference.

Key words: approximate Bayesian computation, deep learner, diffusion models, Ebola, fiducial inference, generative artificial intelligence (AI), normalizing flows, quantile Bayes, reinforcement learning

*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@anl.gov

1 Introduction

This paper reviews recent advances in simulation-based methods for posterior parameter inference, emphasizing approaches that approximate the mapping from data to parameters without explicitly evaluating the likelihood function. Traditional techniques, such as Markov Chain Monte Carlo (MCMC), require a fully specified model and repeated likelihood evaluation, which can be highly computationally prohibitive and often infeasible. While likelihood-free alternatives, such as Approximate Bayesian Computation (ABC), avoid the requirement to specify the likelihood function, they are computationally costly when the outcome space is large. The methods reviewed here take a different approach: they approximate the posterior distribution by learning the relationship between simulated data and the underlying parameters using flexible functions, such as deep learners, built on artificial parameter-outcome pairs. Generative framework is computationally less costly alternative for posterior parameter inference, particularly in cases with many outcomes, nonlinear dependencies, or incomplete model specification.

A fundamental task in modern statistics and machine learning is to construct a predictive mapping from high-dimensional input data to corresponding output labels. Given a collection of input-output pairs $\{(X_i, Y_i)\}_{i=1}^N$, one goal is to build a fast and memory-efficient "look-up" table (or dictionary) for efficient storage, search, and retrieval of data. This can be viewed as a problem of data encoding or compression. A second, more classical problem is that of prediction: *can we find a simple yet powerful prediction rule $f(X_i)$ to evaluate new inputs X_i and predict the corresponding output Y_i*? In this context, we are given training data $\{(Y_i, X_i)\}_{i=1}^N$ and aim to learn a function f that minimizes prediction error. Given observed pairs (Y_i, X_i), the task reduces to supervised nonparametric regression of the form

$$Y_i = f(X_i) + \varepsilon_i, \quad X_i = (X_{i,1}, \ldots, X_{i,d}).$$

For example, in a medical prediction task, each input X_i might represent a vector of patient features such as age, blood pressure, cholesterol levels, and genetic markers, while the output Y_i could indicate whether the patient developed a particular disease within a specified time frame. The goal is to learn a function f that best predicts health outcomes for new patients based on similar covariates.

Beyond prediction, a closely related problem is *inference*: learning not only a point estimate, but a full distribution over model parameters, conditional on the observed data. This brings us to a central task in Bayesian statistics—computing the posterior distribution of unknown parameters (θ_i) given data. Suppose we are given a likelihood function $p(Y_i \mid \theta_i)$, or a forward model $Y_i = f(\theta_i)$, along with a prior distribution $p(\theta_i)$. The aim

is to compute the posterior distribution $p(\theta_i \mid Y_i)$, which quantifies uncertainty over the parameters after observing the outcome(s). For instance, assume we model the probability of developing a disease using logistic regression. The input X_i might represent patient covariates such as age, blood pressure, and cholesterol, while Y_i indicates whether a patient has a disease. Here, the parameter θ_i corresponds to the vector of regression coefficients—each measuring the strength of association between a covariate and the disease risk.

Traditional approaches to Bayesian inference, such as Markov Chain Monte Carlo (Gelfand, 2000; Smith, 2007), estimate the posterior distribution by iteratively sampling from the parameter space and repeatedly evaluating the likelihood function. While asymptotically exact, these methods are computationally expensive and require the full specification of a likelihood, which may be infeasible or misspecified in complex models.

A less restrictive alternative to traditional likelihood-based inference is Approximate Bayesian Computation (ABC) (Sisson et al., 2018). ABC replaces the need for explicit likelihood evaluation with simulation. The method draws candidate parameter values from a prior distribution and simulates outcome conditional on each draw. It retains a parameter if the simulated outcome closely resembles the observed outcome, typically based on a distance between summary statistics. While ABC is useful when the likelihood function is unavailable or intractable, it becomes computationally expensive and inefficient with high-dimensional outcomes, where matching simulated and observed outcomes is more difficult.

In this article, we review methods in variational inference and some of the recent advances. Instead of sampling from the posterior, one learns a deterministic function that maps random noise, typically drawn from a simple distribution such as a standard normal or uniform, to the space of parameters conditional on observed data. This allows for generating approximate posterior samples without explicit likelihood computation. Among such methods, quantile-based approaches learn the conditional distribution of parameters by estimating a collection of posterior quantiles directly. These estimators are model-free unlike normalizing flows or invertible transformations, and are particularly effective in high-dimensional settings.

The goal of our paper is to estimate a deterministic function that transforms a simple random variable into draws from the posterior distribution of the parameters given the observed outcome. This idea is formalized through generative modeling. Let $Z_i \sim p(Z_i)$ be a latent random variable drawn from a fixed reference distribution, such as a multivariate normal or a uniform distribution. The goal is to approximate the conditional distribution $p(\theta_i \mid Y_i)$ using a large sample of training data $\{(Y_i, \theta_i)\}_{i=1}^N \sim p(Y_i, \theta_i)$. A

function f is a deep learner and defined such that $\theta_i = f(Y_i, Z_i)$. A deep learner can then be trained on simulated triples $\{(Y_i, \theta_i, Z_i)\}_{i=1}^N \sim p(Y_i, \theta_i) \times p(Z_i)$. The resulting estimator $\hat{f}_N$ defines a transformation from the reference distribution to the posterior and can be used to approximate posterior draws without explicitly evaluating the likelihood function. When Z_i follows a uniform distribution and θ_i is scalar, this method amounts to learning the inverse conditional quantile function, i.e., $\theta_i = F^{-1}_{\theta|y}(Z_i)$.

Suppose we simulate a large number of training samples $\{(\theta_i, Y_i)\}_{i=1}^N$ from the joint distribution of parameters and data. A classical objective in generative modeling is to estimate the conditional expectation $\hat{\theta}(Y_i) = \mathbb{E}[\theta_i \mid Y_i]$, which minimizes the 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. Traditional approaches to estimating f, such as kernel and nearest-neighbor methods, perform well in low-dimensional settings but scale poorly with complexity. Recent work has explored the use of deep neural networks to approximate f, with growing theoretical understanding of their behavior in high-dimensional and nonparametric regimes; see, e.g., 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 $p(\theta_i \mid Y_i)$, one can extend this framework by constructing a function that maps both the observed data and an independent source of randomness into parameter space. Let Z_i be a latent input drawn from a known reference distribution, and define

$$\theta_i = G(S(Y_i), \psi(Z_i)),$$

where $S(Y_i)$ is a data-dependent summary and $\psi(Z_i)$ is a transformation of the latent input—such as a fixed basis expansion (see e.g. Reid, 1995). The function G, often implemented as a neural network, is trained on simulated pairs (Y_i, θ_i) to approximate the posterior parameter draws conditional on Y_i. Here, the outcome-parameter pairs (Y_i, θ_i) are simulated jointly and the number of simulations is assumed to be large (e.g. $N = 10^9$, where $i = 1, \ldots, N$). Once trained, this procedure enables fast sampling from the posterior without repeated simulation. Because the training sample size N is chosen by the researcher, understanding the statistical properties of the resulting estimators is essential. This includes their generalization error, asymptotic behavior, interpolation properties,

such as the *double descent* phenomenon [1] [2].

The remainder of the paper is structured as follows. Section 2 introduces several architectures for generative modeling. These include approaches based on latent representations, such as auto-encoders (Albert et al., 2022; Akesson et al., 2021), and methods that avoid explicit likelihood evaluation, as in simulation-based or implicit models (Diggle and Gratton, 1984; Baker et al., 2022; Schultz et al., 2022). The framework also connects to the literature on indirect inference, which constructs estimators by aligning features of observed and simulated data rather than maximizing a likelihood function (Pastorello et al., 2003; Stroud et al., 2003; Drovandi et al., 2011, 2015). We review common generative methods:

- Approximate Bayesian computation (ABC) (Rubin, 1984; Beaumont et al., 2002; Blum and François, 2010): likelihood-free inference using summary statistics to match observed and simulated data.
- Variational autoencoders (VAE) (Kingma and Welling, 2014; Kingma et al., 2019): generative models trained to approximate the posterior likelihood fo covariates by optimizing a lower bound on the likelihood.
- Independent component analysis (ICA) (Lee, 1998; Hyvärinen and Oja, 2000; Dinh et al., 2014): dimension reduction model for separating latent sources from observed data.
- Normalizing flows (NF) (Rezende and Mohamed, 2015a) and Invertible neural networks (INN) (Ardizzone et al., 2019): methods that learn invertible transformations from simple reference distributions to complex observed distributions.
- Generative adversarial networks (GAN) (Goodfellow et al., 2014) and conditional GANs (Mirza and Osindero, 2014): models trained through a game-theoretic setup between generator and discriminator networks.
- Deep fiducial inference (DFI) (Fraser, 1961; Hannig et al., 2016).

[1] See Schmidt-Hieber (2020), Shen et al. (2021) and Padilla et al. (2022) for the recent work on interpolation properties of quantile neural networks. See also Belkin et al. (2019); Bach (2024).

[2] Generative inference is closely related to bootstrap methods, which involve simulating datasets, either parametrically or nonparametrically, in order to approximate the sampling distribution of parameter estimates by repeatedly re-estimating the model (see, e.g., Efron, 1982, 1992; Efron and Tibshirani, 1994; DiCiccio and Efron, 1996). In contrast to the bootstrap, generative modeling jointly simulates parameter–outcome pairs and typically requires fitting the model only once. This substantially reduces computational time when compared to bootstrapping.

Section 3 introduces Generative Bayesian computation (GBC), a method for posterior inference based on learning a mapping from observed outcomes and auxiliary randomness to parameter values with a flexible learner. Section 4 presents an empirical application to the Ebola dataset, and Section 5 concludes with suggestions for future research.

2 Generative Modeling

In this section, we describe the idea behind generative inference. Let $Z_i \sim p(Z_i)$ denote a latent random variable drawn from a known base distribution, for example, the uniform distribution between $[0, 1]$ for each observation $i = 1, \ldots, N$. Suppose we have a joint likelihood for observed and latent variables, defined by

$$p_\theta(Y_i, Z_i) = p(Y_i, Z_i \mid \theta_i) = p(Y_i \mid \theta_i)\, p(Z_i),$$

where $\theta_i \sim p(\theta_i)$ is a random parameter drawn from a prior distribution. The goal is to learn a pair of mappings that represent both the forward data-generating process and its inverse:

$$Y_i = f(Z_i) \quad \text{and} \quad Z_i = g(Y_i),$$

where f maps latent variables into the space of observables and g inverts this transformation. Intuitively, if Z_i is a uniform random variable between $[0, 1]$, then f approximates the entire outcome space by mapping quantiles back to the outcome space. f could be thought of as an inverse cumulative distribution function (CDF) that maps the quantiles back to the pre-image defined by the outcome values. Then g transforms these outcomes back to the quantiles. Both functions are superpositions (aka compositions) of simpler functions:

$$f = f_T \circ \cdots \circ f_1 \quad \text{and} \quad g = g_1 \circ \cdots \circ g_T.$$

Each f_t and g_t is typically a low-dimensional, often univariate transformation (e.g., affine followed by a nonlinearity), designed to be easily invertible and to have a tractable Jacobian determinant. [3] The inference amounts to estimating conditional posterior distribution $p(\theta_i \mid Y_i)$. To approximate this distribution, we introduce a generator function $G(Y_i, Z_i)$, which defines an imputed parameter value:

[3] Such transformations form the basis of flow-based generative models, which are widely used to represent probability distributions over Y_i by transforming samples from a simpler latent distribution Z_i. See Papamakarios et al. (2017) for an application of masked autoregressive flows to density estimation.

$$\theta_i = G(Y_i, Z_i), \quad \text{where } Z_i \sim p(z). \tag{1}$$

A diagnostic function $D(Y_i)$ may be used to evaluate whether the generated data distribution matches the marginal distribution of observed data. Under mild regularity conditions, a result often referred to as the noise outsourcing theorem described below guarantees the existence of a generator $G^\star$ such that $\theta_i = G^\star(Y_i, Z_i)$ converges to the true posterior distribution $p(\theta_i \mid Y_i)$. We now apply the noise outsourcing theorem to the problem of calculating a joint distribution Y_i, θ_i and conditional distribution $\theta_i \mid Y_i$, required for generative Bayesian inference.

2.1 Noise Outsourcing Theorem

Let (Y_i, θ_i) be random variables defined on a Borel space $(\mathcal{Y}, \Theta)$. We propose a nonparametric generative approach to constructing a conditional distribution $p(\theta_i \mid Y_i)$. For a given value of the predictor Y_i, we estimate a function $G(Z_i, Y_i)$ where Z_i is a random variable from a reference distribution, e.g. a uniform, such that $G(Z_i, Y_i) \sim p(\theta_i \mid Y_i)$. To sample from $p(Y_i \mid \theta_i)$, we simply generate Z_i from the reference distribution and evaluate $G(Z_i, Y_i)$. This provides a connection between the conditional distribution estimation and the generalized nonparametric regression. We propose to use a quantile neural network to learn the function $G(Z_i, Y_i)$. Specifically, our goal is to find a function $G : \mathbb{R}^d \times \mathcal{Y} \to \Theta$ such that the conditional distribution of $G(Z_i, Y_i)$, given $Y_i = y$ is the same as the conditional distribution of θ_i given $Y_i = y$. Since Z_i is independent of Y_i, it is equivalent of finding a G such that $G(Z_i, Y_i) \sim p(\theta_i \mid Y_i)$, for any $Y_i \in \mathcal{Y}$. The underpinning is the noise outsourcing theorem (Kallenberg, 1997, Theorem 5.10) which states that any random variable can be represented as a function of another random variable and a noise term, namely

$$(Y_i, \theta_i) \overset{a.s.}{=} (Y_i, G^\star(Z_i, Y_i) \sim p(\theta_i \mid Y_i) \text{ for any } Y_i \in \mathcal{Y}.$$

This lemma also provides a unified view of conditional distribution estimation and nonparametric regression. To see this, it is useful to reverse the order and to write

$$\theta_i \mid Y_i = y \sim G^\star(Z_i, y) \text{ for any } y \in \mathcal{Y}.$$

$G^\star : [0, 1] \times \mathcal{Y}$ a measurable (but not necessarily differentiable) function. This shows that the problem of finding $G^\star$ is equivalent to a non-linear non-parametric regression.

This representation, which rewrites the conditional distribution of parameters given data as a deterministic function of observed data and an independent random input, follows from a general result in probability theory is also known as functional representation.

In many practical applications, the observed outcomes are high-dimensional, and we can improve the performance of the generator by using a summary statistic $S(Y_i)$. Here $S : \Re^N \to \Re^k$ is a k-dimensional sufficient statistic, which is a lower-dimensional representation of Y_i. The summary statistic is a sufficient statistic in the Bayes sense (Kolmogorov, 1942). The main idea is to use a generator (a deep neural network here) to approximate the posterior quantiles of the parameter $Q_\pi(\theta_i|Y_i)$ as the optimal estimate of $S(Y_i)$. If there exists a sufficient statistic $S(Y_i)$ that d-separates the outcomes Y_i and parameters θ_i, and i.e. $Y_i \perp \theta_i \mid S(Y_i)$, then the posterior distribution admits the representation

$$\theta_i \mid Y_i \overset{a.s.}{=} G^\star(Z_i, S(Y_i)).$$

In this case, the dependence between outcomes Y_i and parameters θ_i are completely mediated by the sufficient statistic $S(Y_i)$, and the randomness is "outsourced" to Z_i. Intuitively, $S(Y_i)$ contains all relevant information about θ_i reflected in Y_i. When $S(Y_i)$ is not known analytically, one can approximate it using the conditional expectation $G^\star(Z_i, S(Y_i)) = \hat{\mathbb{E}}[\theta_i \mid Y_i]$, which can be estimated nonparametrically using dimensionality reduction models.

2.2 Dimensionality Reduction

We may build summary statistic $\hat{S}(Y_i)$ using techniques such as autoencoders, deep neural networks as shown in Jiang et al. (2017), or partial least squares, as discussed in Polson et al. (2021), building on foundational results by Brillinger (2012) and Bhadra et al. (2021). A comprehensive overview of comparative methods is provided in Blum et al. (2013), and kernel-based embeddings are explored in Park et al. (2016). In the context of likelihood-free inference, generative methods avoid the need for explicit density evaluation required by approaches like MCMC. Instead, they rely on a low-dimensional sufficient statistic and non-linear function approximators to estimate posterior likelihood of parameters. The choice of the summary statistic $\hat{S}(Y_i)$ plays a critical role. In exponential family models, Beaumont et al. (2002) and Nunes and Balding (2010) discuss optimal selection strategies, while smoothing-based extensions of ABC are proposed in Jiang et al. (2018) and Bernton et al. (2019). Additional contributions using basis expansions include Fearnhead and Prangle (2012) and Longstaff and Schwartz (2001), and estimation with latent variables is addressed in Pastorello et al. (2003). Our work builds on Jiang et al. (2017), who pro-

pose using deep neural networks to estimate sufficient statistic and provide asymptotic guarantees for these methods. We also incorporate insights from Dabney et al. (2018) and Ostrovski et al. (2018), who demonstrate that implicit quantile-based neural networks can approximate posterior distributions, particularly in decision-theoretic settings.

A statistic $S(Y_i)$ is sufficient in the Bayesian sense (Kolmogorov, 1942) if, for any prior distribution p, the posterior probability of any measurable set $B \subseteq \Theta$ satisfies

$$p(\theta_i \in B \mid Y_i) = p(\theta_i \in B \mid S(Y_i)).$$

Intuitively, $S(Y_i)$ is sufficient for summarizing information about Y_i relevant for inferring θ_i. In parametric settings, sufficient statistics are often derived directly from the model structure. However, many forward models, particularly those with intractable or implicit likelihoods, we do not have such closed-form sufficient statistics. In such cases, one can approximate sufficient statistics using low-dimensional summary statistics (e.g. feature extraction/selection in a neural network). Accordingly, we assume the existence of a function whose output has dimensionality less than or equal to that of the original outcome variable, and retains the relevant information for the parameter(s) θ_i.

To estimate the posterior map, we simulate training pairs (Y_i, θ_i) from the joint distribution implied by the prior and forward model. For each quantile level q, we define the function

$$f_q(Y_i) := p(\theta_i \leq q \mid Y_i),$$

which corresponds to posterior cumulative probabilities. By choosing a dense grid of quantile levels, we can estimate the full conditional distribution of $\theta_i \mid Y_i$ through interpolation.

A deep neural network can be trained to approximate the posterior quantiles $f_q(Y_i)$ by minimizing the quantile loss over a large simulated dataset $\{(Y_i, \theta_i)\}_{i=1}^{N} \sim p \times \mathcal{M}$, where p denotes the prior distribution and $\mathcal{M}$ represents the forward model. Unlike traditional ABC methods, which rely on rejection sampling [4] to construct approximate posteriors, this approach bypasses that step by evaluating the trained network directly at the observed data $Y_i = y$. The quality of the approximation depends on the smoothness and generalization ability of the neural networks to interpolate from the simulated training pairs to the true data. Appendix A.1 characterizes Bayes risk bounds for such estimators.

[4] ABC draws parameter values from the prior, simulates corresponding outcomes conditional on these parameters from the model, and keeps only those parameter values that generate outcomes close to the observed outcomes, as measured by a chosen summary statistic. This process, known as rejection sampling, discards most simulated draws, particularly in high-dimensional settings, making it computationally expensive.

3 Generative Architectures

We now review existing generative methods that inspire the use of generative artificial intelligence, namely, approximate Bayesian computation (ABC), variational autoencoders (VAE), independent component analysis (ICA), normalizing flows (NF), invertible neural networks (INN), generative adversarial networks (GAN), conditional GANs, and deep fiducial inference (DFI). We focus on empirical understanding of generative architectures and motivating examples.

3.1 Approximate Bayesian Computation

Approximate Bayesian Computation (ABC) is a generative method for conducting Bayesian inference in settings where the likelihood function is intractable or unavailable, but data can be generated from a known structural model. The goal is to approximate the posterior distribution $p(\theta_i \mid Y_i)$ without evaluating the likelihood explicitly. ABC proceeds by simulating a reference table of parameter–data pairs $\{(\theta_i, Y_i^\star)\}_{i=1}^N$, where each $Y_i^\star \sim p(Y_i^\star \mid \theta_i)$, and comparing the simulated output $Y_i^\star$ to the observed data Y_i. To improve computational tractability, both the observed and simulated data are reduced via a low-dimensional summary statistic $S(Y_i) \in \mathbb{R}^k$, and $S(Y_i^\star) \in \mathbb{R}^k$ where $k \ll N$. A common ABC posterior approximation is:

$$p^\epsilon_{\text{ABC}}(\theta_i \mid Y_i) = \frac{1}{m_\epsilon(Y_i)} \int K_\epsilon(S(Y_i^\star) - S(Y_i))\, p(Y_i^\star \mid \theta_i)\, p(\theta_i)\, dY_i^\star, \tag{2}$$

where K_ϵ is a kernel function (e.g., uniform or Gaussian) with bandwidth ϵ, and $m_\epsilon(Y_i)$ is a normalizing constant. As $\epsilon \to 0$ and if $S(Y_i^\star)$ is sufficient for θ_i, this approximation converges to the true posterior:

$$p^\epsilon_{\text{ABC}}(\theta_i \mid Y_i) \to p(\theta_i \mid Y_i). \tag{3}$$

When a uniform kernel is used, the ABC posterior simplifies to a conditional distribution over draws satisfying $\|S(Y_i^\star) - S(Y_i)\| < \epsilon$, i.e.,

$$p^\epsilon_{\text{ABC}}(\theta_i \mid Y_i) \propto p(\theta_i) \cdot p\left(\|S(Y_i^\star) - S(Y_i)\| < \epsilon \mid \theta_i\right).$$

Figure 1 visualizes ABC. Jiang et al. (2017) show that the posterior mean $\mathbb{E}[\theta_i \mid Y_i]$ is an optimal summary statistic under quadratic loss, and propose learning $S(Y_i)$ using deep neural networks. While conceptually appealing, conventional ABC methods suffer

from inefficiency: as ϵ decreases, the approximation improves but the acceptance rate becomes prohibitively low, especially in high-dimensional settings. To address this, recent advances replace the kernel-based approximation with learned transport maps. Specifically, neural networks are trained to map between a baseline distribution (e.g., Gaussian) and the posterior, thus bypassing explicit density estimation. This approach reframes Bayesian inference as a high-dimensional L_2 optimization problem, with the posterior implicitly learned through stochastic gradient descent. Such frameworks form the foundation of modern generative inference methods, including diffusion-based Bayesian samplers.

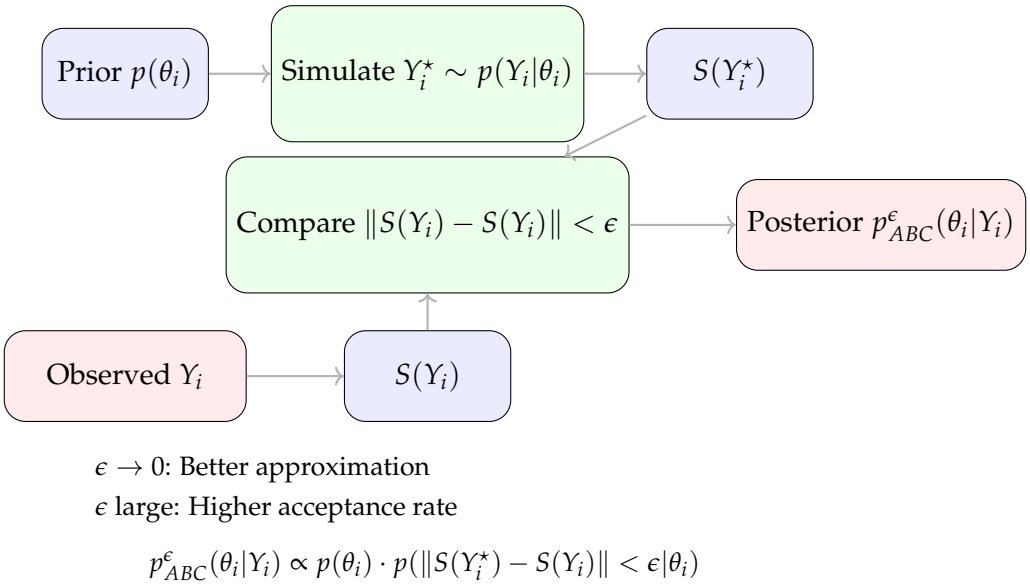


Figure 1: Approximate Bayesian Computation (ABC) framework.

Example. A central problem in population genetics is modeling historical demographic processes, such as migration, population bottlenecks, or natural selection, from observed genetic data. Traditional likelihood-based inference is computationally intensive due to the complexity of evolutionary models and high-dimensional genotype spaces. Generative methods, particularly approximate Bayesian computation (ABC), are a simulation-based alternatives. By simulating genetic data under competing demographic scenarios and comparing summary statistics, such as allele frequencies or linkage disequilibrium patterns, to observed data, ABC enables researchers to model evolutionary history without explicitly defining the likelihood function of the data.

3.2 Variational Autoencoders

Let $X_i \in \mathcal{X}$ denote an observed data point, such as an image, a voice recording, or any complex high-dimensional data. The magic of a variational autoencoder (VAE) lies in its ability to not just compress this data, but to learn how to generate new, similar data points. Assume, we wish to understand and create new faces. Instead of memorizing every pixel, the VAE learns to represent faces using a set of meaningful latent variables $Z_i \in \mathcal{Z}$ — features like smile width, eye shape, or hair style. This compressed representation (the latent space) is not just for storage; it's a creative space where we can "mix and match" features to generate entirely new, realistic faces that never existed before. The advantage of VAEs is that they learn both how to break down complex data into these meaningful features (encoding) and how to reconstruct convincing new examples from these features (decoding). This makes them powerful tools for creative tasks like generating new images, synthesizing speech, or even designing new molecules — anywhere we want to create new examples that share the essential characteristics of our training data. Figure 2 shows that VAE reduces dimensionality, and generates new similar data points.

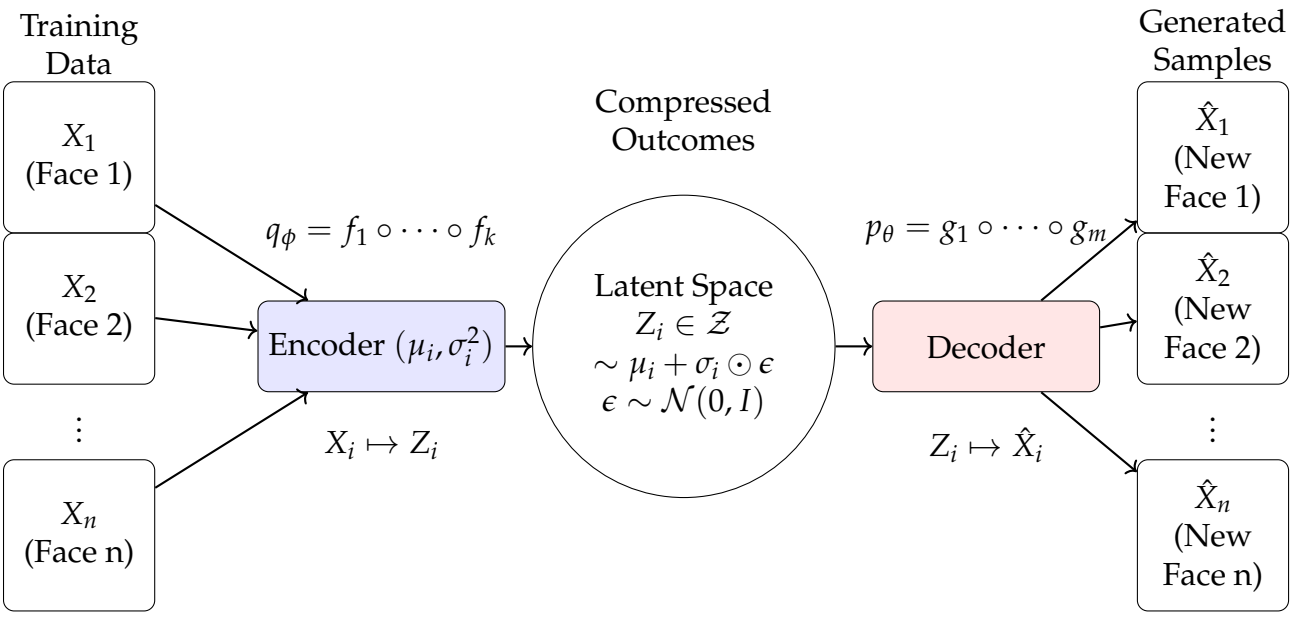


Figure 2: Variational Autoencoders. The encoder (q_ϕ) and decoder (p_θ) are composite functions that map between data space (X_i) and latent space (Z_i). The encoder compresses input data into a meaningful latent representation, from which the decoder generates new, similar examples.

The underlying generative model is defined in two parts. A prior distribution over latent variables

$$Z_i \sim p(Z_i) = \mathcal{N}(0, I),$$

representing mean-zero, independent unobservables; and a conditional model of the outcome given the latent variable:

$$X_i \mid Z_i \sim p_\theta(X_i \mid Z_i),$$

where θ indexes parameters of a flexible function (e.g., a neural network).

The goal of the encoder is to calculate the posterior likelihood of latent variables conditional on covariates:

$$p_\theta(Z_i \mid X_i) = \frac{p_\theta(X_i \mid Z_i) \cdot p(Z_i)}{p_\theta(X_i)},$$

The marginal likelihood of the observed data is then given by integrating out the latent variable:

$$p_\theta(X_i) = \int p_\theta(X_i \mid Z_i)\, p(Z_i)\, dZ_i. \tag{4}$$

Latent Space This integral is generally intractable due to the complexity of the conditional model $p_\theta(X_i \mid Z_i)$. Rather than attempting to compute the intractable posterior $p_\theta(Z_i \mid X_i)$ directly, the VAE introduces a variational approximation:

$$q_\phi(Z_i \mid X_i) = \mathcal{N}(Z_i \mid \mu_i, \text{diag}(\sigma_i^2)),$$

where $\mu_i = \mu_\phi(X_i)$ and $\sigma_i = \sigma_\phi(X_i)$ are outputs of an encoder network parameterized by ϕ. This step can be interpreted as projecting the observed outcome X_i onto a distribution over unobservables, analogous to estimating a reduced-form distribution over latent variables.

Sampling and Reconstruction To generate latent representations, the VAE uses the reparameterization trick:

$$Z_i = \mu_i + \sigma_i \odot \epsilon, \quad \epsilon \sim \mathcal{N}(0, I),$$

which expresses sampling from $q_\phi(Z_i \mid X_i)$ as a differentiable function of μ_i, σ_i, and a standard normal noise vector. This permits optimization via gradient descent. The sampled latent variable Z_i is then passed through the decoder to produce a reconstruction:

$$\hat{X}_i = f_\theta(Z_i),$$

where f_θ is the composite function estimated by the decoder network.

Objective Function: The ELBO Starting from the (intractable) marginal likelihood

$$\log p_\theta(X_i) = \log \int p_\theta(X_i \mid Z_i)\, p(Z_i)\, dZ_i, \tag{5}$$

we insert an arbitrary variational density $q_\phi(Z_i \mid X_i)$ that integrates to one:

$$\begin{aligned} p_\theta(X_i) &= \int \frac{p_\theta(X_i, Z_i)}{q_\phi(Z_i \mid X_i)}\, q_\phi(Z_i \mid X_i)\, dZ_i \\ &= \mathbb{E}_{q_\phi(Z_i|X_i)}\Big[\tfrac{p_\theta(X_i,Z_i)}{q_\phi(Z_i|X_i)}\Big]. \end{aligned} \tag{6}$$

Applying Jensen's inequality to the concave function log,

$$\begin{aligned} \log p_\theta(X_i) &= \log \mathbb{E}_{q_\phi}\Big[\tfrac{p_\theta(X_i,Z_i)}{q_\phi(Z_i|X_i)}\Big] \\ &\geq \mathbb{E}_{q_\phi(Z_i|X_i)}\Big[\log \tfrac{p_\theta(X_i,Z_i)}{q_\phi(Z_i|X_i)}\Big] \;=\; \mathcal{L}(\theta, \phi; X_i). \end{aligned} \tag{7}$$

By Bayes' rule (i.e. the definition of a joint density),

$$p_\theta(X_i, Z_i) = p_\theta(X_i \mid Z_i)\, p(Z_i),$$

so inside the logarithm we have

$$\log \frac{p_\theta(X_i, Z_i)}{q_\phi(Z_i \mid X_i)} = \underbrace{\log p_\theta(X_i \mid Z_i)}_{\text{reconstruction}} + \underbrace{\log p(Z_i)}_{\text{prior}} - \underbrace{\log q_\phi(Z_i \mid X_i)}_{\text{entropy term}}.$$

Taking expectations under $q_\phi(Z_i \mid X_i)$ yields the familiar evidence lower bound (ELBO):

$$\begin{aligned} \mathcal{L}(\theta, \phi; X_i) &= \mathbb{E}_{q_\phi}\big[\log p_\theta(X_i \mid Z_i)\big] - \mathbb{E}_{q_\phi}\big[\log q_\phi(Z_i \mid X_i)\big] + \mathbb{E}_{q_\phi}\big[\log p(Z_i)\big] \\ &= \mathbb{E}_{q_\phi}\big[\log p_\theta(X_i \mid Z_i)\big] - \mathrm{KL}\big(q_\phi(Z_i \mid X_i) \,\|\, p(Z_i)\big), \end{aligned} \tag{8}$$

where $\mathrm{KL}(\cdot\|\cdot)$ denotes the Kullback–Leibler divergence. Maximising the ELBO thus (i) tightens the lower bound on $\log p_\theta(X_i)$ and (ii) drives $q_\phi(Z_i \mid X_i)$ towards the true posterior $p_\theta(Z_i \mid X_i)$.

Example: Household Consumption. Suppose X_i represents a vector of observed household consumption expenditures across multiple categories (e.g., food, housing, transportation), and Z_i denotes unobserved preference parameters that govern household demand. The encoder maps observed consumption patterns into a distribution over latent preferences, measuring household-specific heterogeneity in tastes and constraints. The

decoder reconstructs the consumption bundle from a given realization of these latent preferences. This setup allows the model to learn a flexible, nonlinear demand system that can generalize across households with different unobserved characteristics. Sampling $Z_i \sim q_\phi(Z_i \mid X_i)$ enables posterior predictive simulations of consumption responses to changes in prices or income, akin to counterfactual simulations in structural demand models. In this interpretation, the encoder approximates the posterior distribution of latent utility parameters conditional on observed consumption, while the decoder functions as a structural mapping from latent preferences to predicted behavior. See Heaton et al. (2016) who discuss deep learners and variational autoencoders for applications in Finance, such as predicting stock prices, or default probability.

Example: Medical Images. In medical imaging, the scarcity of annotated data and the constraints imposed by patient privacy limit the applicability of traditional supervised learning models. Generative models such as variational autoencoders (VAEs) address this by learning low-dimensional latent representations of complex medical images, such as MRI or CT scans, while also modeling uncertainty present in the data. This model generates synthetic images that mimic rare pathologies or underrepresented patient populations, improving the robustness of downstream prediction tasks.

3.3 Independent Component Analysis

Independent Component Analysis (ICA) can be viewed as a latent variable model, as shown in MacKay (1999). The basic setup assumes that the observed covariate $X_i \in \mathbb{R}^K$ is generated as a linear mixture of unobserved, statistically independent components $Z_i \in \mathbb{R}^K$. This corresponds to the structural relationship:

$$X_i = AZ_i,$$

where A is an unknown invertible mixing matrix. The objective in ICA is to recover the inverse matrix $W = A^{-1}$, such that the latent sources can be recovered via $Z_i = WX_i$ (see Figure 3).

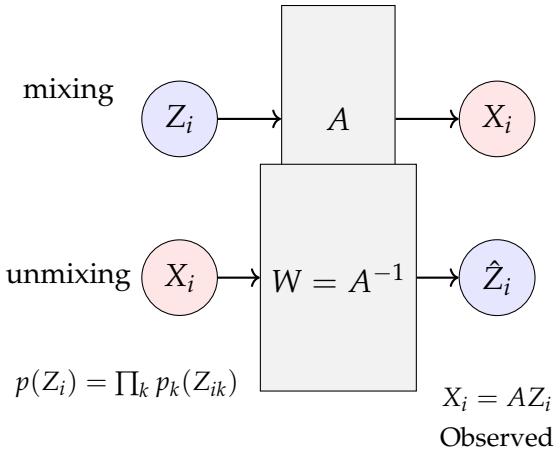


Figure 3: Linear Independent Component Analysis.

The latent components Z_i are assumed to be mutually independent:

$$p(Z_i) = \prod_{k=1}^{K} p_k(Z_{ik}),$$

where each p_k is typically chosen to be non-Gaussian. This non-Gaussianity assumption is essential for the identifiability of W; if Z_i were multivariate Gaussian, the model would be rotationally invariant and hence non-identifiable. The joint model over the observed and latent variables is given by:

$$p(X_i, Z_i \mid A) = \underbrace{\delta(X_i - AZ_i)}_{\text{enforces } X_i = AZ_i} \cdot p(Z_i),$$

where $\delta(X_i - AZ_i)$ is a Dirac delta distribution (also called the unit impulse) and it imposes the deterministic mapping between latent and observed variables. Inference can proceed in two equivalent ways: 1) Treat the model as a latent variable model and estimate parameters using maximum likelihood or Bayesian methods. 2) Marginalize out Z_i analytically and optimize the marginal likelihood of the observed data:

$$p(X_i \mid A) = \int \delta(X_i - AZ_i)\, p(Z_i)\, dZ_i = \frac{1}{|\det A|} \prod_{k=1}^{K} p_k\left((A^{-1}X_i)_k\right),$$

where we use the identity $\int \delta(X_i - AZ_i) f(Z_i)\, dZ_i = |\det A|^{-1} f(A^{-1}X_i)$.

Proof. We use the multivariate scaling rule of the Dirac delta,

$$\delta(Mu) = \frac{\delta(u)}{|\det M|}, \qquad M \in \mathbb{R}^{K\times K} \text{ invertible},$$

which generalises the one–dimensional identity $\delta(ax) = \delta(x)/|a|$. Because A is invertible, write $X_i - AZ_i = A\big(A^{-1}X_i - Z_i\big)$; applying the rule gives

$$\delta(X_i - AZ_i) = \frac{\delta\big(Z_i - A^{-1}X_i\big)}{|\det A|}.$$

Hence

$$\int_{\mathbb{R}^K} \delta(X_i - AZ_i)\, f(Z_i)\, dZ_i = \frac{1}{|\det A|} \int_{\mathbb{R}^K} \delta\big(Z_i - A^{-1}X_i\big)\, f(Z_i)\, dZ_i.$$

The remaining delta collapses the integral to the point $Z_i = A^{-1}X_i$, yielding the identity

$$\int \delta(X_i - AZ_i)\, f(Z_i)\, dZ_i = \frac{1}{|\det A|}\, f\big(A^{-1}X_i\big)\ .$$

Taking $f(Z_i) = p(Z_i) = \prod_{k=1}^{K} p_k(Z_{ik})$ reproduces the closed-form marginal likelihood $p(X_i \mid A) = |\det A|^{-1} \prod_{k=1}^{K} p_k\big((A^{-1}X_i)_k\big)$ used in ICA. □

This formulation shows that the independent component analysis is a special case of a latent variable model with a deterministic and linear "decoder" $X_i = AZ_i$, and an independent prior over latent variables.

Dinh et al. (2015) introduce a deep learning framework for high-dimensional density estimation by learning an invertible, stable mapping between observed covariates $X_i \sim p_X$ and latent variables $Z_i \sim p_Z$. This framework was later extended by the Real NVP method (Dinh et al., 2017), which improves stability and scalability of such bijective transformations. Normalizing flows based on these ideas have been applied to conditional density estimation (Trippe and Turner, 2018) and complex posterior approximations (Rezende and Mohamed, 2015a). Building on nonlinear independent component estimation, Müller et al. (2019) propose the neural importance sampler, where proposal densities for Monte Carlo integration are parametrized by neural networks.

Example: Financial Data. A major challenge in financial data is the separation of underlying structural components from observed financial signals, which are noisy and interdependent. Independent component analysis (ICA) provides a generative framework for disentangling latent, statistically independent sources from asset returns or macroeconomic indicators. For example, in equity markets, ICA can identify unobserved risk

factors, such as liquidity shocks or sector-specific dynamics, that drive co-movements in asset prices. These decompositions are valuable for portfolio optimization, risk attribution, and detecting market anomalies, particularly when structural identification is difficult using standard factor models.

3.4 Normalizing Flows

Let $X_i \in \mathbb{R}^n$ be a random variable with unknown density $p_X(x)$. Normalizing flows model X_i as a transformation $G : \mathbb{R}^n \to \mathbb{R}^n$ of a latent variable $Z_i \in \mathbb{R}^n$ with known density $p_Z(z)$, such that $X_i = G(Z_i)$. The key idea is to design G to be invertible with a tractable Jacobian determinant, which enables exact density evaluation via the change-of-variables formula:

$$p_X(x) = p_Z(z) \cdot |\det J_G(z)|^{-1}, \quad \text{where} \quad z = G^{-1}(x), \tag{9}$$

and $J_G(z) = \partial G(z)/\partial z$ is the Jacobian of G. The invertibility requirement ensures that sampling ($X_i = G(Z_i)$) and density evaluation ($Z_i = G^{-1}(X_i)$) are both computationally feasible. To construct flexible yet tractable transformations, G is decomposed into a composition of K simpler bijective functions $T_k : \mathbb{R}^n \to \mathbb{R}^n$:

$$G = T_K \circ T_{K-1} \circ \cdots \circ T_1. \tag{10}$$

Let $z_0 = z$ and $z_K = x$. The forward pass computes x from z via successive transformations:

$$z_k = T_k(z_{k-1}), \quad k = 1, \ldots, K, \tag{11}$$

while the inverse pass recovers z from x:

$$z_{k-1} = T_k^{-1}(z_k), \quad k = K, \ldots, 1. \tag{12}$$

The Jacobian determinant of G factors into a product over the transformations $\{T_k\}$ due to the chain rule:

$$\det J_G(z) = \prod_{k=1}^{K} \det \left(\frac{\partial T_k(z_{k-1})}{\partial z_{k-1}} \right). \tag{13}$$

This decomposition allows the log-likelihood of observed data $\{X_i\}_{i=1}^N$ to be expressed as:

$$\log p_X(X_i) = \log p_Z(G^{-1}(X_i)) - \log \left| \det J_G(G^{-1}(X_i)) \right|, \tag{14}$$

which is maximized to estimate the parameters of G. The choice of transformations $\{T_k\}$

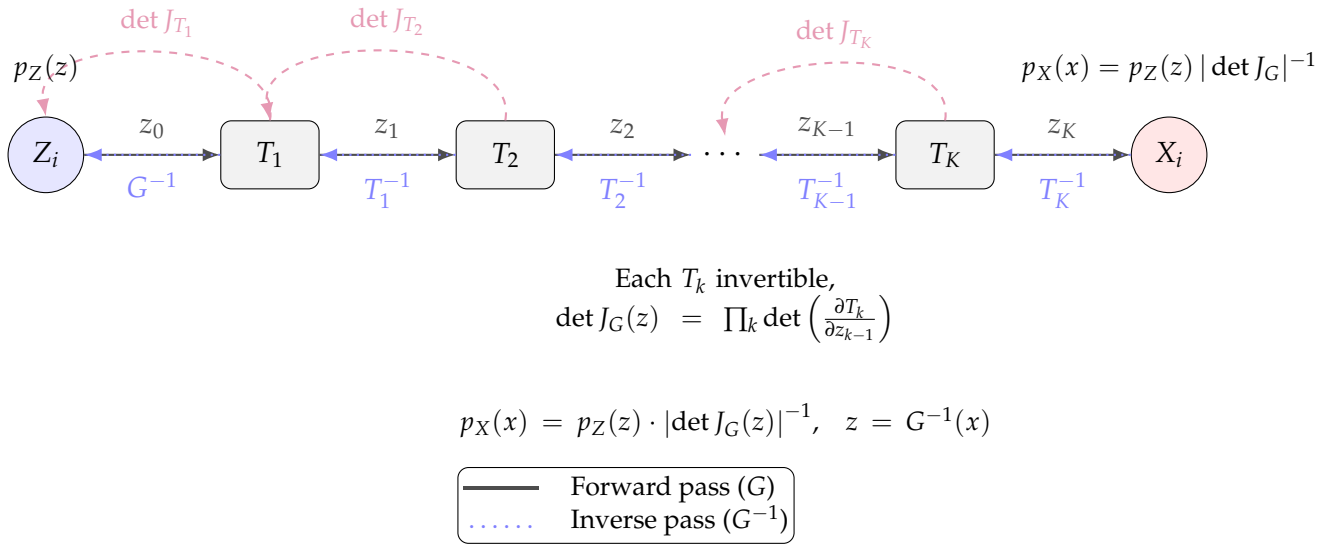


Figure 4: Normalizing Flows.

balances expressiveness and computational efficiency, with common options including affine couplings and invertible linear maps.

Example: Anomaly Detection in Network Traffic Data Normalizing flows is a powerful tool for modeling complex distributions such as network traffic data. Suppose $X_i \in \mathbb{R}^n$ represents summary statistics of network connections, including features like packet counts, durations, and byte transfers. To identify anomalies, we aim to model the distribution of normal traffic behavior. Using a normalizing flow, we assume a latent variable $Z_i \sim \mathcal{N}(0, I_n)$ and learn an invertible transformation G such that $X_i = G(Z_i)$. By training G to maximize the exact log-likelihood of the observed traffic data, we end up with an explicit density estimator $p_X(x)$. After training, the likelihood of new observations $p_X(x_{\text{new}})$ can be derived efficiently. Traffic instances with low likelihood under the learned model are flagged as potential anomalies, enabling a principled, probabilistic approach to anomaly detection without requiring explicit labels (see Polson and Sokolov (2017) for traffic flow prediction with a deep learner).

Example: Molecular Structure Predicting molecular structure and interaction dynamics involves navigating highly multimodal energy landscapes governed by physical laws, such as the Boltzmann distribution. Traditional simulation methods often struggle with efficient sampling in such spaces, particularly when rare conformations are of interest. Normalizing flows (NF) address this challenge by learning invertible transformations from simple reference distributions to the complex target distributions of molecular con-

figurations. In applications such as protein–ligand binding, NFs can model binding affinities and predict structural modes without perturbing equilibrium properties of the system.

Flow Transformation Models. Let $X_i \in \mathbb{R}^n$ and $Y_i \in \mathbb{R}^n$ denote random variables. A flow-based transformation specifies a mapping $Y_i = f(X_i)$, where $f : \mathbb{R}^N \to \mathbb{R}^n$ is an invertible, differentiable function. Typically, f is modeled as an invertible neural network designed to ensure that the determinant of its Jacobian is tractable. Given a latent density $p_F(\cdot)$ and a conditional density $p_Y(\cdot \mid \cdot, s)$, the joint distribution of (Y_i, X_i) conditional on covariates s satisfies

$$\begin{aligned} p(Y_i, X_i \mid s) &= p(Y_i \mid X_i, s)\, p(X_i \mid s) \\ &= p_Y(Y_i \mid f^{-1}(X_i), s)\, p_F(f^{-1}(X_i)) \left| \det \left(\frac{\partial f^{-1}(X_i)}{\partial X_i} \right) \right|. \end{aligned}$$

Here, the Jacobian matrix $\partial f^{-1}(X_i)/\partial X_i$ accounts for the change of variables under the transformation f, and its determinant adjusts the probability mass accordingly. Flow-based models can be interpreted as a form of nonlinear latent factor models, where observed variables are generated by deterministic transformations of independent latent factors. Applications include extensions of independent component analysis to nonlinear settings, where the dimensionality of the latent representation matches that of the observed data; see Camuto et al. (2021) for theoretical developments in this direction.

Latent Factor Models with Flow Transformations. A special class of normalizing flow models incorporates latent factors through invertible neural networks (INNs). Suppose (Y_i, X_i) denote observed random variables, and (F_i, S_i) represent latent factors. The model is specified as

$$Y_i = F_i^\top S_i + \varepsilon_i, \tag{15}$$

$$X_i = f(F_i), \tag{16}$$

$$F_i \sim \mathcal{N}(0, I_p), \quad S_i \sim \mathcal{N}(0, I_s), \tag{17}$$

where $f : \mathbb{R}^p \to \mathbb{R}^p$ is an invertible, differentiable function, and ε_i is mean-zero Gaussian noise independent of (F_i, S_i). In this formulation, the latent factors F_i are transformed through f to generate the observed covariates X_i, while Y_i depends linearly on F_i and S_i.

Estimation proceeds by solving a joint optimization problem over f and S_i. The objec-

tive function takes the form

$$L(f, \{S_i\}_{i=1}^N) = \sum_{i=1}^N \left\{ \lambda \|Y_i - f^{-1}(X_i)^\top S_i\|^2 + \|f^{-1}(X_i)\|^2 - \log \left| \det \left(\frac{\partial f^{-1}}{\partial X_i} \right) \right| \right\} - \sum_{i=1}^N \log p(S_i),$$

where $\lambda > 0$ is a regularization parameter. The first term enforces the linear relationship between Y_i and the reconstructed latent factors $F_i = f^{-1}(X_i)$, while the second and third terms correspond to the latent prior density and the Jacobian adjustment from the flow transformation.

An iterative two-step procedure can be used to minimize $L(f, \{S_i\}_{i=1}^N)$ (with $\{S_i\}_{i=1}^N = \{S_i\}$):

1. Update the flow map: $\hat{f}^{(t)} = \hat{f}^{(t-1)} - \eta \nabla_h L(f, \{\hat{S}_i\}^{(t-1)})$, where η is the learning rate.

2. Update the latent states: $\hat{S}_i^{(t)} = \arg\min_S L(\hat{f}^{(t)}, \{S_i\})$, or alternatively, sample from the posterior distribution proportional to $\exp\left(-L(\hat{f}^{(t)}, \{S_i\})\right)$.

Invertible Neural Network: An important concept in the context of generative models is an invertible neural network or INNs (Dinh et al., 2017). Loosely speaking, an INN is a one-to-one function with a forward mapping $f : \mathbb{R}^d \mapsto \mathbb{R}^d$, and its inverse $g = f^{-1}$. Song et al. (2019) provide the 'MintNet' algorithm to construct INNs by using simple building blocks of triangular matrices, leading to efficient and exact Jacobian calculation. On the other hand, Behrmann et al. (2021) show that common INN methods suffer from exploding inverses and provide conditions for stability of INNs. For image representation, Jacobsen et al. (2018) introduce a deep invertible network, called the i-revnet, that retains all information from input data up until the final layer. HINT (Hierarchical Invertible Neural Transport) networks are discussed by Kruse et al. (2021) who provide the algorithm for posterior sampling. In this formulation, the function T moves in the normalizing direction: a complicated and irregular data distribution $p_W(W_i)$ towards the simpler, more regular or 'normal' form of the base measure $p_Z(Z_i)$.

3.5 Generative Adversarial Networks

The third class of methods we review are generative adversarial networks, proposed by Goodfellow et al. (2020). Here we learn the implicit probability distribution over parameters $\theta = \{\theta_g, \theta_d\}$ by defining a deterministic map $X_i^{\text{gen}} = G_{\theta_g}(Z_i, X_i)$, called generator.

The basic idea of GAN is to introduce a nuisance neural network $D_{\theta_d}(X_i)$, called discriminator and parameterised by θ_d and then jointly estimate the parameters θ_g of the generator function $G_{\theta_g}(Z_i, X_i)$ and the discriminator. The discriminator network is a binary classifier which is trained to discriminate generated and real samples X_i. The network parameters are found by minimizing standard binomial log-likelihood (a.k.a cross-entropy)

$$J(\theta_d, \theta_g) = -\frac{1}{2}\mathbb{E}_x[\log D_{\theta_d}(x)] - \frac{1}{2}\mathbb{E}_z[\log(1 - D_{\theta_d}(G_{\theta_g}(z, x))]. \tag{18}$$

To calculate the first term, we just use empirical expectation calculated using observed training samples. Next, we need to specify the cost function for the generator function. Assuming a zero-sum scenario in which the sum of the cost for generator and discriminator is zero, we use the mini-max estimator, which jointly estimates the parameters θ_d (and θ_g as a by-product) and is defined as follows:

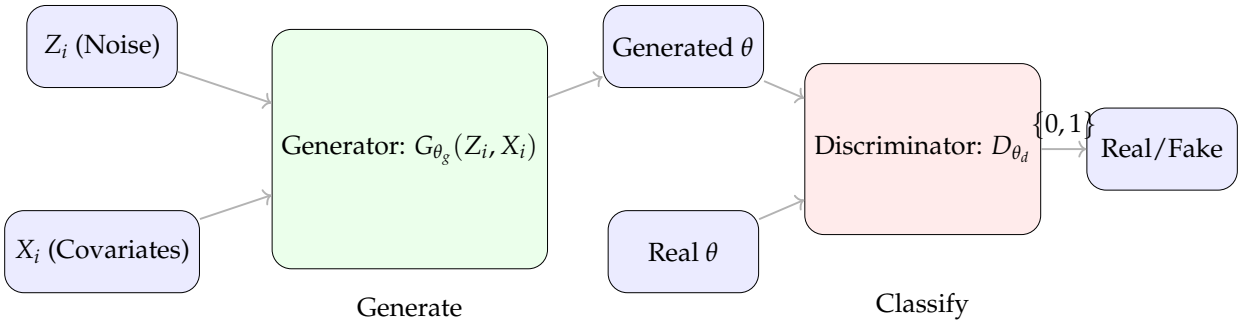
$$\min_{\theta_g} \max_{\theta_d} J(\theta_d, \theta_g) \tag{19}$$

The term adversarial, which is misleading, was used due to the analogy with game theory. In GANs the generator network tries to "trick" the discriminator network by generating samples that cannot be distinguished from real samples available from the training data set.

In conditional GAN (cGAN) architectures, we generate a reference table from $X_i^{\text{gen}} = G(W_i, Z_i)$ where W_i represent covariates (features). In this case, we have to be able to decide when the marginals for the data match up. To do so, we use the Jensen-Shannon Kullback-Leibler divergence to compare the distributions $p_{obs} = p(x_{obs})$ and $p_\theta = p(x)$. This can be expressed as

$$JS(p_\theta^{(n)}, p_{\theta_0}^{(n)}) = \ln 2 + \frac{1}{2} \sup_{D:\mathcal{X}\to[0,1]} \left(E_{x\sim p_{obs}}(\ln D(x)) + E_{x\sim p_\theta}(\ln(1 - D(x)))\right)$$

This leads to a minimax game interpretation and the need to find two neural networks, a generator G and a discriminator D. The discriminator plays the same role as the $\epsilon-$neighborhood in ABC simulation as a way of deciding whether $p(x_{obs})$ is close to $p(x)$.



$$\min_{\theta_g} \max_{\theta_d} J(\theta_d, \theta_g)$$

$$J(\theta_d, \theta_g) = -\tfrac{1}{2} E_x[\log D_{\theta_d}(X_i)] - \tfrac{1}{2} E_z[\log(1 - D_{\theta_d}(G_{\theta_g}(Z_i, X_i)))]$$

Figure 5: Generative Adversarial Networks.

Example: Patient Outcomes Consider a medical study where we observe patient outcomes Y_i under different treatments Z_i, along with their characteristics X_i (e.g., age, biomarkers). A GAN learns to generate synthetic outcomes $\hat{Y}_i = G_{\theta_g}(Z_i, X_i)$ by transforming random noise $Z_i \sim \mathcal{N}(0, I)$ and patient features X_i into plausible treatment responses. The discriminator D_{θ_d} then evaluates whether an outcome (real or generated) is consistent with the patient's profile and treatment assignment. Through this adversarial process, the generator learns to produce synthetic outcomes with known counterfactual treatment effects. For instance, given a new patient, we can generate multiple synthetic outcomes by sampling different noise vectors, providing a distribution of potential treatment effects that helps quantify uncertainty in treatment decisions.

Example: Images for Charity Donation. Athey et al. (2022) investigate how seemingly minor stylistic choices in profile images, such as smiling or wearing sunglasses, affect user outcomes and contribute to disparities on online platforms, using Kiva, a micro-lending site, as a case study. The authors distinguish between fixed (or type) characteristics like gender or age and modifiable (or style) features like smiles or photo framing. They conduct experiments with GAN-generated profile variants to investigate the impact of these features. The authors fund that style attributes like smiling boost funding chances, and that women and younger users tend to adopt more positively perceived styles.

3.6 Diffusion Models

Diffusion models fall into this class as they consist of a forward simulation process, a reverse process and a sampling procedure. They are designed to transport samples to a base distribution, typically Gaussian. Here $X_0 \sim q(X)$ leads to a forward diffusion pro-

cess $q(X_t|X_{t-1})$. Similar to stochastic gradient Langevin diffusion MCMC (and bridge sampling). Diffusion Model consists of a forward process (or diffusion process), in which a datum (generally an image) is progressively noised, and a reverse process (or reverse diffusion process), in which noise is transformed back into a sample from the target distribution. The sampling chain transitions in the forward process can be set to conditional Gaussians when the noise level is sufficiently low. Combining this fact with the Markov assumption leads to a simple parameterization of the forward process:

$$q(X_T|X_0) = \prod_{t=0}^{T} q(X_t|X_{t-1}) = \prod_{t=0}^{T} N(\sqrt{1-\beta_t}X_{t-1}, \sqrt{\beta_t})$$

where β_t is a variance schedule (either learned or fixed).

Example. Consider the task of generating high-resolution facial images. In this case, X_0 represents real face images drawn from the data distribution. The forward process gradually corrupts these images by adding Gaussian noise at each step. The model is trained to learn how to reverse this corruption, effectively denoising samples drawn from the base Gaussian distribution. At generation time, one starts from pure noise $X_T \sim \mathcal{N}(0, I)$, and iteratively applies the learned reverse denoising steps to produce a synthetic face image $\hat{X}_0 \sim q(X_T)$. Compared to GANs, diffusion models often yield higher fidelity samples with greater stability during training, especially with high-dimensional covariates.

3.7 Deep Fiducial Inference

Fiducial inference is a likelihood-based framework that constructs a data-dependent distribution over parameters without requiring a prior (Hannig et al., 2016). Suppose we observe a sequence of random variables $Y_1, \ldots, Y_N$, generated according to the structural model

$$Y_i = G(U_i, \theta),$$

where G is a known deterministic function, θ is the unknown parameter of interest, and U_i are latent random variables with known distribution F_U. If the function G is invertible in θ for fixed U_i, then for each realization $Y_i = y$, one can construct a *fiducial map* $\theta = Q_{y_i}(U_i) := G^{-1}(y, U_i)$. This induces a distribution over θ by propagating the randomness of U_i through the inverse map.

A canonical example is the additive model $Y_i = \theta + U_i$, where $U_i \sim \mathcal{N}(0, 1)$. In this case, the fiducial distribution is $\theta \sim \mathcal{N}(Y_i, 1)$. For i.i.d. data $Y_i \sim \mathcal{N}(\mu, \sigma^2)$, the sufficient

statistic—the sample mean and sample variance—can be written as:

$$\bar{Y} = \mu + \sigma U_1, \quad S^2 = \sigma^2 U_2, \quad U_1 \sim \mathcal{N}\left(0, \tfrac{1}{n}\right), \quad U_2 \sim \text{Gamma}\left(\tfrac{n-1}{2}, \tfrac{n-1}{2}\right).$$

Inverting these relationships provides the fiducial distribution for μ and σ, with $\mu \sim \mathcal{N}(\bar{Y}, \sigma^2/n)$ conditional on σ.

Ronald Fisher also proposed deriving fiducial distributions by inverting confidence intervals (i.e., quantile functions). Define $\theta_Y(\alpha)$ to be the α-quantile such that

$$p_Y(\theta \leq \theta_Y(\alpha)) = \alpha.$$

Then the cumulative distribution function of the fiducial distribution is $\alpha_Y(\theta) := p_Y(\theta' \leq \theta)$, and the corresponding fiducial density is given by:

$$p(\theta \mid Y_i) := \frac{d}{d\theta}\alpha_Y(\theta).$$

Although no prior is explicitly specified, the resulting distribution often resembles the Bayesian posterior under Jeffreys' prior, particularly in regular parametric models. This arises naturally when transforming variables via the inverse of the data-generating process:

$$f_Y(Y_i \mid \theta) = f_U(G^{-1}(Y_i, \theta)) \cdot \left|\det\left(\frac{\partial G^{-1}(Y_i, \theta)}{\partial Y_i}\right)\right|,$$

where the Jacobian term encodes curvature and connects to the Fisher information. In this way, fiducial inference implicitly incorporates an information-based prior structure. One may wish to design fiducial rejection sampling algorithm. To draw approximate samples from the fiducial posterior for observation Y_i:

1. Simulate $U_i^\star \sim F_U$.
2. Solve for $\theta_i^\star = \arg\min_\theta \|Y_i - G(U_i^\star, \theta)\|$.
3. Compute $Y_i^\star = G(U_i^\star, \theta_i^\star)$.
4. Accept $\theta_i^\star$ if $\|Y_i - Y_i^\star\| \leq \epsilon$.

This algorithm approximates the posterior $p(\theta_i \mid Y_i)$ as $\epsilon \to 0$. Efficiency can be improved by replacing Y_i with a lower-dimensional summary statistic $S(Y_i)$, and accepting if $\|S(Y_i) - S(Y_i^\star)\| \leq \epsilon$.

4 Generative Bayesian Computation

To fix notation, let $\mathcal{Y}$ denote a locally compact metric space representing the space of observable signals, with associated Borel σ-algebra $\mathcal{B}(\mathcal{Y})$. Let λ be a reference measure defined on the measurable space $(\mathcal{Y}, \mathcal{B}(\mathcal{Y}))$. Let τ denote a quantile level, corresponding to a realization of a reference random variable $Z_i \sim p(Z_i)$, where $p(Z_i)$ is typically taken to be the uniform distribution on $[0, 1]$ as described earlier. For each value of an unobserved parameter θ, the conditional distribution of the signal $Y_i \in \mathcal{Y}$ is denoted by $p(dy \mid \theta)$. Let Θ be a locally compact metric space of latent parameters (such as hidden states or structural variables), with Borel σ-algebra $\mathcal{B}(\Theta)$. A measure μ is defined on $(\Theta, \mathcal{B}(\Theta))$. The conditional distribution of $\theta \in \Theta$ given an observed signal $y \in \mathcal{Y}$, that is, the posterior distribution, is denoted by $p(d\theta \mid y)$.

When absolutely continuous with respect to μ, the posterior admits a density $p(\theta \mid y)$ such that

$$p(d\theta \mid y) = p(\theta \mid y)\, \mu(d\theta).$$

Similarly, the prior distribution on θ is denoted $p(d\theta) = p(\theta)\, \mu(d\theta)$, when a density p with respect to μ exists.

Our framework accommodates both likelihood-based and likelihood-free models for inference. In likelihood-free settings, the data-generating process is characterized by a deterministic mapping,

$$y = f(\theta),$$

where f denotes a forward operator linking parameters θ to observed signals y. When a likelihood function $p(y \mid \theta)$ exists with respect to a reference measure λ, the conditional distribution is represented as

$$p(dy \mid \theta) = p(y \mid \theta)\, \lambda(dy).$$

A central advantage of likelihood-free methods is their ability to bypass density evaluation by relying solely on simulation. This permits the use of flexible generative mechanisms, including deep neural networks, to approximate the posterior distribution from simulated data. In practice, we approximate the inverse posterior map $\theta_i = F^{-1}_{\theta|y}(\tau)$ using deep neural networks, where τ is a vector of uniform random variables (i.e. quantiles here). For vector-valued parameters, we model the joint posterior via an autoregressive factorization:

$$\theta_i = \left(F^{-1}_{\theta_1}(\tau_1),\ F^{-1}_{\theta_2|\theta_1}(\tau_2),\ \ldots \right).$$

Our proposed architecture uses a ReLU-based neural network to parameterize the posterior quantile map. Crucially, the first layer of the network incorporates the utility function, allowing the method to prioritize regions of the parameter space that contribute most to expected utility. This contrasts with traditional two-stage approaches that first learn the posterior and then apply Monte Carlo integration—an approach that may be inefficient in the presence of utility functions with mass in the posterior tails. Appendix A.2 describes quantile methods as alternatives to density computation.

4.1 Quantile Reordering in Reinforcement Learning

Standard approaches to sequential decision-making, including reinforcement learning, typically estimate the expected discounted utility of future outcomes, commonly referred to as the value function:

$$V(s,d) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t u(x_t) \,\middle|\, s_0 = s, d_0 = d\right], \tag{20}$$

where s denotes the initial state, d the initial decision, x_t the outcome at time t, $u(x_t)$ the instantaneous utility derived from x_t, and $\gamma \in (0,1)$ a discount factor reflecting time preferences. This formulation, while analytically convenient, reduces the entire distribution of future utilities to their mean, thereby discarding important information about risk, variability, and the probability of extreme (e.g., low-utility) outcomes.

To address this limitation, Dabney et al. (2017) propose modeling the entire distribution of returns. This distributional perspective is particularly relevant in contexts where the shape of the outcome distribution, rather than just its mean, affects the quality of a decision, such as in finance, labor markets, or dynamic treatments. A central tool in this approach is the quantile function. Let $U \sim p_U$ denote a random utility or return variable. The expectation of U can be represented in terms of its quantile function F_U^{-1} as:

$$\mathbb{E}[U] = \int_0^1 F_U^{-1}(\tau)\, d\tau. \tag{21}$$

This identity motivates approximating the utility distribution by discretizing the integral using a fixed grid of quantile levels:

$$\tau_i = \frac{2i-1}{2N}, \quad i = 1, \ldots, N, \tag{22}$$

and averaging the associated quantile values. In practice, a neural network is trained to

estimate these quantiles $\hat{F}_U^{-1}(\tau_i \mid s, d)$ conditional on state s and decision d. The outputs are sorted during training to enforce monotonicity:

$$\hat{F}_U^{-1}(\tau_1 \mid s, d) \leq \hat{F}_U^{-1}(\tau_2 \mid s, d) \leq \cdots \leq \hat{F}_U^{-1}(\tau_N \mid s, d). \tag{23}$$

Here, τ_i denotes a quantile level, corresponding to a realization of a reference random variable $Z_i \sim p_Z$, where p_Z is typically taken to be the uniform distribution on $[0, 1]$ described earlier.

4.2 Quantile Neural Networks and Normalizing Flows

Posterior distributions can be naturally represented using the inverse cumulative distribution function (CDF), following the classical von Neumann transformation:

$$\theta_i = F_{\theta_i|Y_i}^{-1}(\tau), \quad \text{where } \tau \sim \text{Unif}(0, 1).$$

This formulation fits within a general inverse-mapping framework of the form:

$$\theta_i = G(S(Y_i), \tau),$$

where $F_{\theta_i|Y_i}^{-1} = G \circ S$, and $S(Y_i)$ denotes a data-dependent sufficient transformation. This motivates the use of quantile-based deep neural networks (DNNs) as a natural class of models for approximating posterior distributions. While standard activation functions such as ReLU or Tanh are commonly used, quantile networks directly reflect the inverse-distributional structure of Bayesian inference. The set of posterior distributions $p(\theta_i \mid Y_i)$ is thus characterized by the identity:

$$\theta_i \overset{D}{=} G(S(Y_i), \tau), \quad \text{where } \tau \sim \text{Unif}(0, 1).$$

The Kolmogorov–Arnold representation theorem ensures that any multivariate continuous function can be expressed as a superposition of univariate functions. As a result, sufficiently deep and wide neural networks can approximate complex mappings from observed data Y_i to latent parameters θ_i.

A second class of methods in the machine learning literature approximates target distributions by learning inverse cumulative distribution functions, or, more generally, by representing the distribution of θ_i through marginalization over an auxiliary latent variable Z_i (Kingma and Welling, 2022). When modeling the inverse CDF directly, the latent variable is often assumed to be uniformly distributed on $(0, 1)$, as in the quantile represen-

tation of Bond-Taylor et al. (2022). One prominent approach in this class is normalizing flows, which define a flexible, deterministic, and invertible transformation of a simple base distribution into the target distribution. Specifically, a sample θ is obtained via

$$\theta_i = G(Z_i, X_i; \theta_{ig}), \quad z \sim p(z),$$

where G is a parameterized function (e.g., a neural network) and θ_{ig} denotes its parameters. If G is invertible and differentiable, the resulting density $p(\theta_i \mid X_i)$ is obtained through the change-of-variables formula:

$$p(\theta_i \mid X_i) = p(Z_i) \left| \det \left(\frac{\partial G^{-1}(\theta_i, X_i)}{\partial \theta} \right) \right|,$$

as shown by Rezende and Mohamed (2015b).

4.3 Deep Quantile Bayes: Fourier Series Representation

Dabney et al. (2018) introduce the implicit quantile network (IQN), a deep learning architecture designed to approximate the full quantile function of a conditional distribution. The IQN estimates $F^{-1}(\tau, X_i)$, the quantile function of a random variable Y_i given predictors X_i, where $\tau \in (0, 1)$ is a quantile index. This representation allows one to generate samples from the conditional distribution $p(Y_i \mid X_i)$ by drawing $\tau \sim \text{Unif}(0, 1)$ and evaluating the network at (τ, X_i).

In our framework, we adopt a related approach by parameterizing the quantile function as a single composite function $F^{-1}(\tau, X_i) = f(\tau, X_i; \theta_i)$. Following Dabney et al. (2018), we represent this function as a composition:

$$F^{-1}(\tau, X_i) = f(\tau, X_i; \theta_i) = g\left(\psi(X_i) \circ \phi(\tau)\right),$$

where g, ψ, and ϕ are feedforward neural networks, and $\circ$ denotes element-wise (Hadamard) multiplication. The quantile embedding $\phi(\tau)$ is constructed using a cosine basis followed by a ReLU activation:

$$\phi_j(\tau) = \text{ReLU}\left(\sum_{i=0}^{n-1} \cos(\pi i \tau) w_{ij} + b_j \right),$$

which is a flexible and smooth encoding of the quantile index. In this architecture, we can learn heteroskedastic and non-Gaussian distributions without requiring explicit like-

lihood specification. The architecture, however, assumes sampling sufficiently large training dataset to overcome overfitting.

4.4 Normal–Normal Bayesian Learning and Wang Distortion.

To illustrate how quantile representations can replace density-based calculations in Bayesian inference, we consider the standard normal–normal model. This formulation also reveals a connection to *Wang's distortion measure*, commonly used in risk-sensitive decision theory, which naturally emerges from the transformation between prior and posterior distributions.

Suppose we observe $Y = (Y_1, \ldots, Y_n)$ with

$$\begin{aligned} Y_1, \ldots, Y_n \mid \theta &\sim \mathcal{N}(\theta, \sigma^2), \\ \theta &\sim \mathcal{N}(\mu, \alpha^2). \end{aligned}$$

The sufficient statistic is the sample mean $\bar{Y} = \frac{1}{n}\sum_{i=1}^{n} Y_i$. Given observations $y = (Y_1, \ldots, Y_n)$, the posterior distribution of $\theta \mid Y$ is normal with updated parameters:

$$\theta \mid Y \sim \mathcal{N}(\mu_*, \sigma_*^2), \quad \mu_* = \frac{\sigma^2\mu + \alpha^2 s}{t}, \quad \sigma_*^2 = \frac{\alpha^2\sigma^2}{t},$$

where $s = \sum_{i=1}^{n} Y_i$ and $t = \sigma^2 + n\alpha^2$.

Rather than working with densities directly, we observe that the posterior and prior CDFs are related through a monotonic transformation:

$$1 - \Phi\left(\frac{\theta - \mu_*}{\sigma_*}\right) = g\left(1 - \Phi\left(\frac{\theta - \mu}{\alpha}\right)\right),$$

where Φ denotes the standard normal CDF. The function g is known as the *Wang distortion function*, given by:

$$g(p) = \Phi\left(\lambda_1 \Phi^{-1}(p) + \lambda\right),$$

with distortion parameters

$$\lambda_1 = \frac{\alpha}{\sigma_*}, \quad \lambda = \frac{\alpha\lambda_1(s - n\mu)}{t}.$$

This expression shows that posterior updating can be viewed as a transformed quantile operation applied to the prior. A direct computation confirms that the distorted prior

CDF matches the posterior CDF:

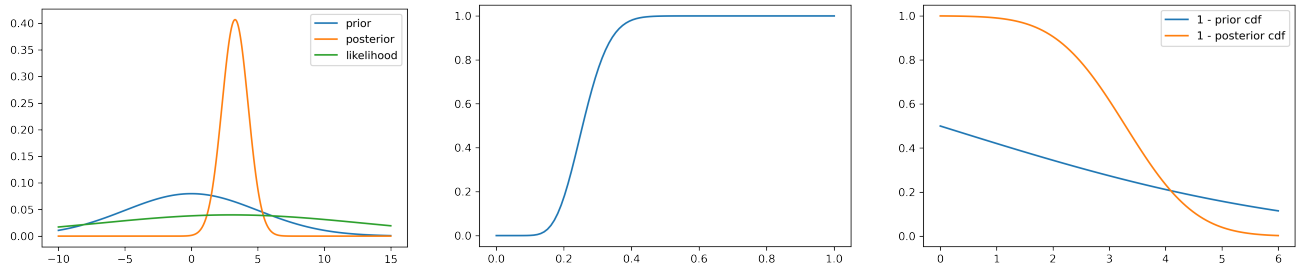
$$g\left(1-\Phi\left(\frac{\theta-\mu}{\alpha}\right)\right)=\Phi\left(\lambda_1\left(-\frac{\theta-\mu}{\alpha}\right)+\lambda\right)=1-\Phi\left(\frac{\theta-\mu_*}{\sigma_*}\right).$$

Thus, the updated parameters satisfy:

$$\sigma_*=\frac{\alpha}{\lambda_1},\quad \mu_*=\mu+\frac{\alpha\lambda}{\lambda_1},$$

where the expression for λ ensures consistency with the sample information. This result represents a new perspective on Bayesian updating, as a distortion of prior quantiles.

Numerical Example Consider the normal-normal model with Prior $\theta \sim N(0,5)$ and likelihood $y \sim N(3,10)$. We generate $n = 100$ samples from the likelihood and calculate the posterior distribution.



(a) Model for simulated data (b) Distortion Function g (c) 1 - Φ

Figure 6: Density for prior, likelihood and posterior, distortion function and 1 - Φ for the prior and posterior of the normal-normal model.

The posterior distribution calculated from the sample is then $\theta \mid y \sim N(3.28, 0.98)$. Figure 6 shows the Wang distortion function for the normal-normal model. The left panel shows the model for the simulated data, while the middle panel shows the distortion function, the right panel shows the 1 - Φ for the prior and posterior of the normal-normal model.

5 Ebola Outbreak

For illustration, we use the multi-output agent-based epidemic model problem documented in Fadikar et al. (2018). We predict the 56-week, simulated outputs for three holdout scenarios.

After the 2014-2015 West Africa Ebola outbreak, the Research and Policy for Infectious Disease Dynamics (RAPIDD) program at the National Institutes of Health (NIH) convened a workshop to compile and explore the various forecasting approaches used to help manage the outbreak. At its conclusion, a disease forecasting challenge was launched to provide 4 synthetic population datasets and scenarios as a baseline for cross-assessment. A stochastic, agent-based model (Ajelli et al., 2018) first generated each population using varying degrees of data accuracy, availability, and intervention measures; individuals were then assigned activities based on demographic and survey data to model realistic disease propagation. Transmission by an infected individual is determined probabilistically based on the duration of contact with a susceptible individual and $d = 5$ static inputs $\Theta = \theta_1, \cdots, \theta_5$ to the model.

Parameter	Description	Range
θ_1	probability of disease transmission	$[3 \times 10^{-5}, 8 \times 10^{-5}]$
θ_2	initial number of infected individuals	$[1, 20]$
θ_3	delay in hospital intervention	$[2, 10]$
θ_4	efficacy of hospital intervention	$[0.1, 0.8]$
θ_5	intervention reduction of travel	$[3 \times 10^{-5}, 8 \times 10^{-5}]$

Figure 7: 5 static inputs used for defining disease propagation for the Ebola ABM

A single run outputs a cumulative count of infected individuals over a 56-week period. For more details on the model and challenge, see Viboud et al. (2018). To maintain comparison, we use the same data set from Fadikar et al. (2018), which consists of a collection of $m = 100$ scenarios generated through a space-filling, symmetric Latin hypercube design. For each scenario, 100 replicates were run for a total of $N = 10,000$ simulated epidemic trajectories. A single run for each parameter set produced a 56-dimensional output, capturing the cumulative weekly number of infected individuals. The log results for every setting combination is depicted in Figure 8.

Notably, different parameter settings produced strongly divergent behaviors in their replicates. Some followed a mean trajectory while others produced strong bimodal behavior, significant heteroskedasticity, or simply remained flat-lined. Fadikar et al. (2018) reasons that only a subset of replicates within each parameter settings may produce similar initial behaviors and that adding an additional variable α to index these replicates would produce more accurate predictions. Modifying the Quantile Kriging approach, the 100 replicates of each parameter setting are replaced with $n_\alpha = 5$ quantile-based trajectories. The quantiles are then indexed within each parameter set by the addition of a

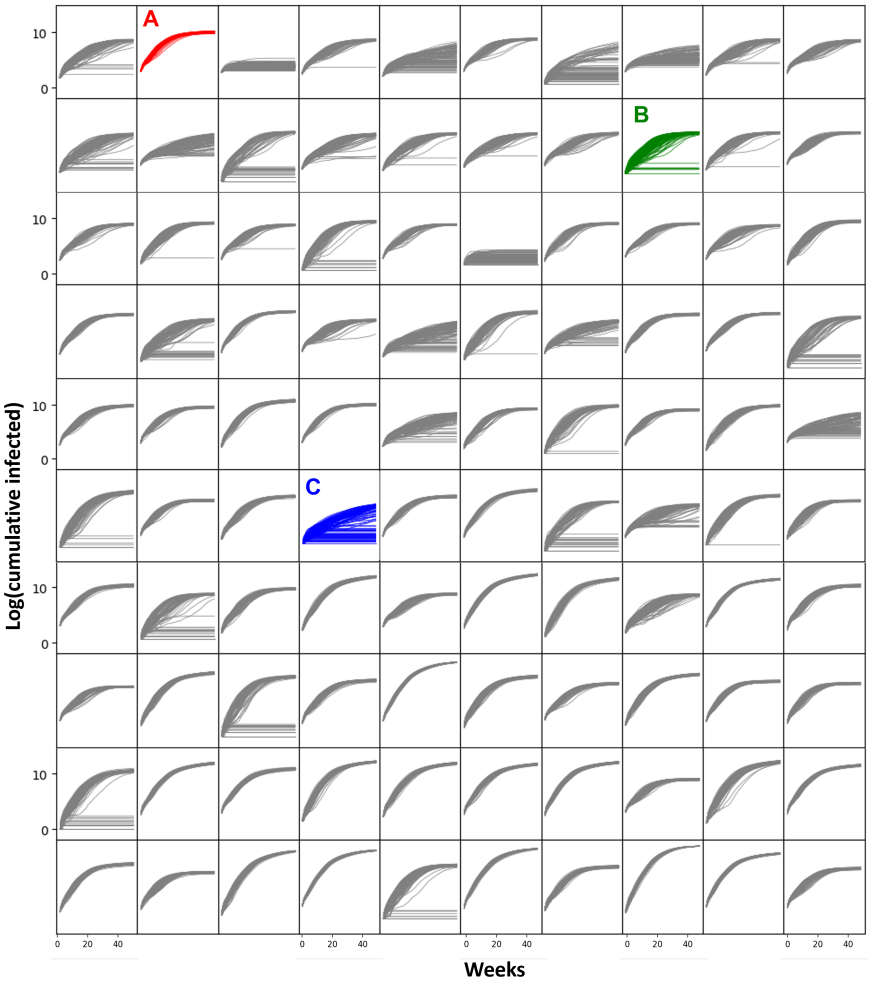


Figure 8: For each scenario, the 100 simulated trajectories for the cumulative number of disease incidences across 56 weeks are shown as grey lines. The three holdout scenarios (A, B, and C) are highlighted in red, green, and blue respectively.

sixth latent variable $\alpha \in [0, 1]$:

$$\Theta = [\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \alpha] \,. \tag{24}$$

For testing the forecast models, 3 unique parameter settings, which we will refer to as A, B, and C, and their respective $n = 100$ simulated outputs were excluded from the training set. Figure 9 shows the 5 calculated quantiles of these three hold-out scenarios, labeled A, B, and C and highlighted in red, green, and blue respectively, that we wish to predict using our model.

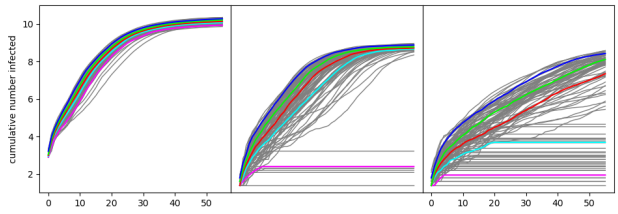


Figure 9: For each holdout scenario, labeled A, B, and C, the 100 simulated trajectories of the cumulative number of disease incidences over 56 weeks are shown as grey lines. The 5 colored lines represent the $[0.05, 0.275, 0.5, 0.725, 0.95]$ quantiles, which are now indexed by the additional sixth parameter α.

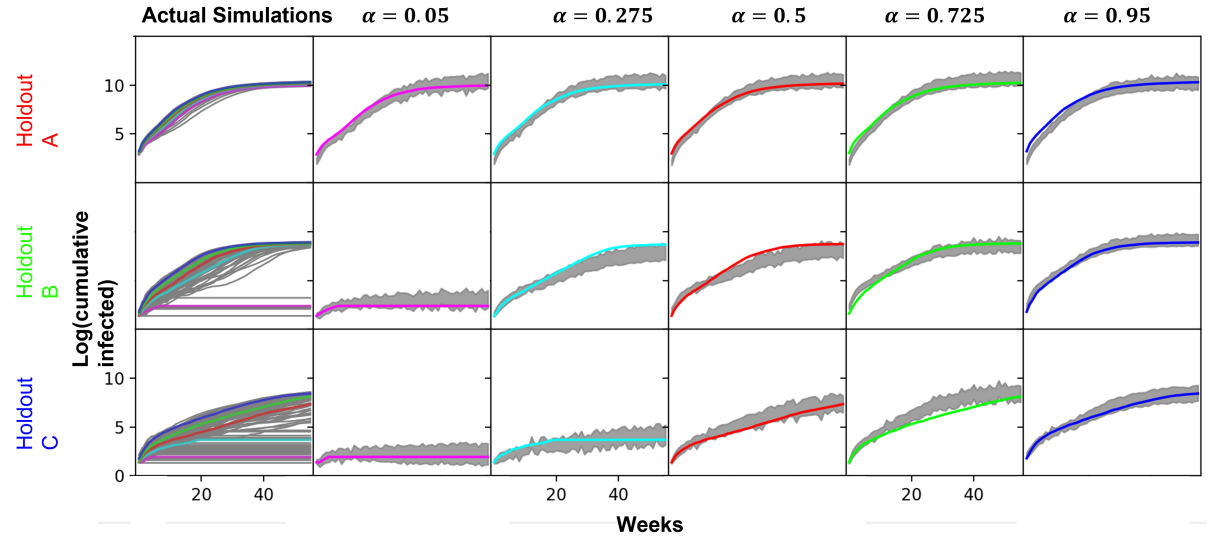
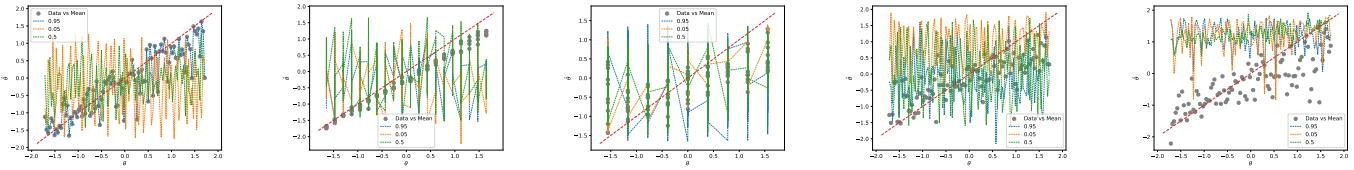


Figure 10: The first column shows the 100 actual simulations and their empirical quantiles for each of the three holdout scenarios. The next 5 columns show the posterior prediction's 90% confidence intervals for each empirical quantile, denoted by different colors.

The model's predicted trajectories are shown in Figure 10. Each row of the figure corresponds to one of the holdout scenarios (A, B, and C), which were not used to train the GP emulator. The first column shows the 100 replicates generated by the ABM, along with the estimated quantiles. The remaining 5 columns compare the 90% confidence intervals our model predicts for each of the 5 quantile settings estimated from the ABM replicates.

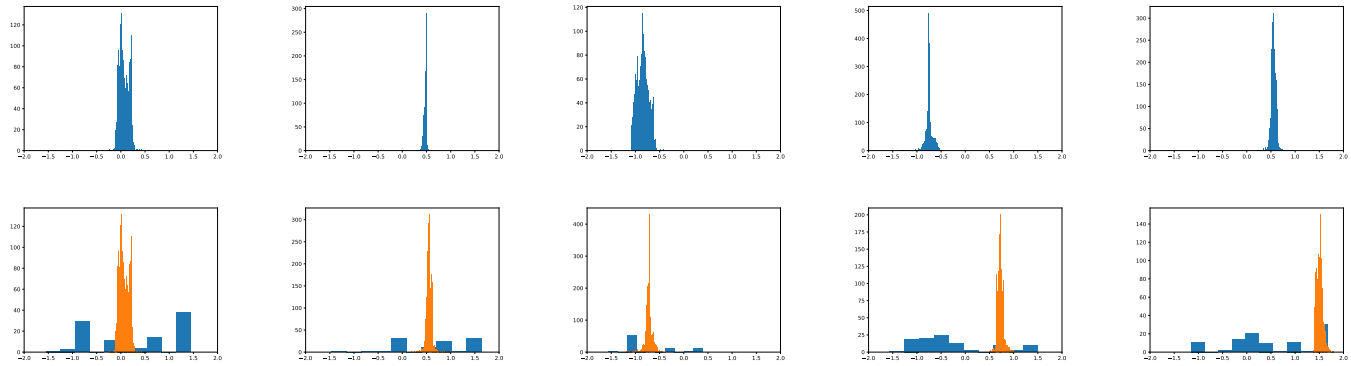
Overall, our model's predictions among the lower quantiles provide more noticeable improvements, including the correction of several of Fadikar et al. (2018) cases, to maintain the quantile across the entire timeline's 90% CI. In contrast, some of our results underestimated the initial case loads when Fadikar et al. (2018)'s model did not; however,

most of these cases incorporated the difference within the following weeks without missing the quantile's inflection points.



θ_1 θ_2 θ_3 θ_4 θ_5

Figure 11: Prediction for 20 randomly selected y's



θ_1 θ_2 θ_3 θ_4 θ_5

Figure 12: Posterior histograms for $y^{(7080)}$

6 Discussion

Generative artificial intelligence refers to a class of simulation-based inference methods that approximate the relationship between parameters and observed data by learning from joint samples of (θ_i, Y_i). These methods apply nonparametric regression, typically implemented with deep neural networks, to estimate a function h that maps dimensionally reduced sufficient statistics of Y_i and exogenous stochastic input (e.g., uniform noise) to the corresponding parameter θ_i. In its simplest form, h corresponds to the inverse conditional quantile function. Once trained, the model can generate approximate posterior samples by evaluating h at observed data and newly drawn random noise, thereby avoiding explicit likelihood function or Markov Chain Monte Carlo. While inherently flexible, open challenges remain in designing architectures capable of incorporating complex parameter dependencies. Addressing these challenges presents a promising direction for

future research in simulation-based, likelihood-free inference.

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.

Ardizzone, L., Lüth, C., Kruse, J., Rother, C., and Köthe, U. (2019). Guided image generation with conditional invertible neural networks. *arXiv preprint arXiv:1907.02392*.

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*.

Blum, M. G. and François, O. (2010). Non-linear regression models for approximate bayesian computation. *Statistics and computing*, 20:63–73.

Blum, M. G. B., Nunes, M. A., Prangle, D., and Sisson, S. A. (2013). A Comparative Review of Dimension Reduction Methods in Approximate Bayesian Computation. *Statistical Science*, 28(2):189–208.

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.

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.

Camuto, A., Willetts, M., Roberts, S., Holmes, C., and Rainforth, T. (2021). Towards a Theoretical Understanding of the Robustness of Variational Autoencoders. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*, pages 3565–3573. PMLR.

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

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. (2014). Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516*.

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.

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.

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.

Fearnhead, P. and Prangle, D. (2012). Constructing summary statistics for approximate Bayesian computation: Semi-automatic approximate Bayesian computation. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 74(3):419–474.

Fraser, D. A. (1961). On fiducial inference. *The Annals of Mathematical Statistics*, 32(3):661–676.

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

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.

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.

Heaton, J., Polson, N. G., and Witte, J. H. (2016). Deep learning in finance. *arXiv preprint arXiv:1602.06561*.

Hyvärinen, A. and Oja, E. (2000). Independent component analysis: algorithms and applications. *Neural networks*, 13(4-5):411–430.

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

Jacobsen, J.-H., Smeulders, A., and Oyallon, E. (2018). I-revnet: Deep invertible networks. *arXiv preprint arXiv:1802.07088*.

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.

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.

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.

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

Lee, T.-W. (1998). Independent component analysis. In *Independent component analysis: Theory and applications*, pages 27–66. Springer.

Longstaff, F. A. and Schwartz, E. S. (2001). Valuing American options by simulation: A simple least-squares approach. *The review of financial studies*, 14(1):113–147.

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

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.

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., Pavlakou, T., and Murray, I. (2017). Masked Autoregressive Flow for Density Estimation. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

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.

Polson, N. G. and Sokolov, V. O. (2017). Deep learning for short-term traffic flow prediction. *Transportation Research Part C: Emerging Technologies*, 79:1–17.

Reid, N. (1995). The roles of conditioning in inference. *Statistical Science*, 10(2):138–157.

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*.

Rubin, D. B. (1984). Bayesianly justifiable and relevant frequency calculations for the applied statistician. *The Annals of Statistics*, pages 1151–1172.

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*.

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

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.

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

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.

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

Viboud, C., Sun, K., Gaffey, R., Ajelli, M., Fumanelli, L., Merler, S., Zhang, Q., Chowell, G., Simonsen, L., and Vespignani, A. (2018). The RAPIDD ebola forecasting challenge: Synthesis and lessons learnt. *Epidemics*, 22:13–21.

A Appendix

A.1 Bayes Risk

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 distribution under a Bayesian prior, resulting in a regularized maximum a posteriori (MAP)

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 $\mathcal{F}$.

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, including ones that do no depend on d and in this sense we avoid the curse of dimensionality. For example, it is common to consider the class of linear superpositions (a.k.a. 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 $z \in \mathbb{R}$,
- $g_1(x_1, \ldots, x_d) = (f_{01}(x_1), \ldots, f_{0d}(x_d))$,
- $g_2(y_1, \ldots, y_d) = \sum_{p=1}^{d} y_p$.

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 ReLU networks achieve the optimal rate $O(N^{-1/3})$ under similar assumptions; Coppejans (2004) finds a convergence rate of $O(N^{-3/7} = O(N^{-3/(2\times 3+1)})$ for threee-times differentiable (cubic B-splines); Igelnik and Parikh (2003) derives a parametric rate $O(N^{-1})$ for Kolmogorov spline networks.

A.2 Bayes Rule for Quantiles

Parzen (2004) shows that quantile methods are direct alternatives to density estimation. 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 identity

$$\theta \overset{P}{=} Q_\theta(F_\theta(\theta)),$$

highlighting that the quantile function can be used to generate draws from the target distribution via transformation of uniform random variables. This property enables efficient sampling schemes by exploiting the monotonicity of the quantile map and aligning order statistics of the parameter and baseline distributions. 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:

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

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