entropy

MDPI

Article

Generative AI for Bayesian Computation

Nick Polson [1] and Vadim Sokolov [2,*]

[1] Booth School of Business, University of Chicago, Chicago, IL 60637, USA; ngp@chicagobooth.edu
[2] Department of Systems Engineering and Operations Research, George Mason University, Fairfax, VA 22030, USA
* Correspondence: vsokolov@gmu.edu

Abstract

Generative Bayesian Computation (GBC) provides a simulation-based approach to Bayesian inference. A Quantile Neural Network (QNN) is trained to map samples from a base distribution to the posterior distribution. Our method applies equally to parametric and likelihood-free models. By generating a large training dataset of parameter–output pairs inference is recast as a supervised learning problem of non-parametric regression. Generative quantile methods have a number of advantages over traditional approaches such as approximate Bayesian computation (ABC) or GANs. Primarily, quantile architectures are density-free and exploit feature selection using dimensionality reducing summary statistics. To illustrate our methodology, we analyze the classic normal–normal learning model and apply it to two real data problems, modeling traffic speed and building a surrogate model for a satellite drag dataset. We compare our methodology to state-of-the-art approaches. Finally, we conclude with directions for future research.

Keywords: generative AI; Bayesian computation; quantile neural networks; traffic flow; satellite drag

1. Introduction

Generative Bayesian Computation (GBC) provides a simulation-based approach for generating samples from a posterior distribution. To do this, we train a deep neural network to map samples from a base distribution to the posterior distribution, thus avoiding the use of densities. The inverse posterior mapping is learned directly via optimization of parameters of a deep generative quantile map. GBC applies to all forms of inference including both likelihood-free and parametric models for predictions and maximum expected utility analysis [1]. To illustrate our methodology, we provide three examples. First, learning in the normal–normal model and, secondly, we build a surrogate model for a satellite drag dataset (see [2]).

The key idea behind GBC is to start with a large training sample from the joint distribution of observable and parameters, denoted by $(y^{(i)}, \theta^{(i)})_{i=1}^{N}$ where training sample size N is large, typically $N \gg n$, $y = (y_1, \ldots, y_n)$, $\theta = (\theta_1, \ldots, \theta_d)$. For the data-generating process, we allow for both parameter likelihood inference $y^{(i)} \mid \theta^{(i)} \sim p(y \mid \theta^{(i)})$ and likelihood-free models where $y^{(i)} = f(\theta^{(i)})$ is a given forward map.

Samples are generated from the prior $\theta^{(i)} \sim \pi(\theta)$, and a base distribution $\tau^{(i)} \sim p(\tau)$, typically a vector of uniforms. We can directly find the mapping

$$\theta^{(i)} = G\left(S\left(y^{(i)}\right), \tau^{(i)}\right),$$

Academic Editor: Ali Mohammad-Djafari

Received: 25 March 2025
Revised: 15 May 2025
Accepted: 12 June 2025
Published: 26 June 2025

Citation: Polson, N.; Sokolov, V. Generative AI for Bayesian Computation. *Entropy* **2025**, *27*, 683. https://doi.org/10.3390/e27070683

Here, S is a summary statistic, which is a lower-dimensional representation of y. The summary statistic is a sufficient statistic in the Bayes sense [3], and G is a generator map to transport the base variable τ to the posterior density $\theta|y$. Both G and S are deep neural networks estimated from the training dataset. In the univariate case, when the base distribution is uniform, the inverse posterior map is simply given by the inverse posterior cumulative distribution function, namely

$$\theta \stackrel{D}{=} F^{-1}_{\theta|y}(S(y), \tau).$$

Here, S is a dimension reducing summary statistic, [4], $y \in R^n$, and $\theta \in R^k$. In the multi-parameter case, when $k > 1$, we use an RNN or autoregressive structure where we model a vector via a sequence

$$(F^{-1}_{\theta_1|y}(\tau_1), F^{-1}_{\theta_2|\theta_1,y}(\tau_2) \ \ldots),$$

and estimate the quantile functions recursively.

Our work then builds on [4] who were the first to propose deep learners for dimension reduction methods for $S(y)$ and to provide asymptotic theoretical results. We also build on the insight by [5,6] that implicit quantile neural networks can be used to approximated distributions that arise in reinforcement learning. Ref. [7] also provides the connection between the widely used 1-Wasserstein distance and quantile regression.

GBC allows the researcher to learn a dimensionality-reduced summary (a.k.a. sufficient) statistics along with a non-linear map [4,8]. A useful interpretation of the sufficient statistic is as a posterior mean, which also allows us to view posterior inputs as one of the inputs to the posterior mean. One can also view a NN is an approximate non-linear Bayes filter to perform such tasks [9].

Our framework provides a natural link for black box methods and stochastic methods, as commonly known in the machine learning literature [10,11]. Quantile deep neural networks and their ReLU/tanh counterparts provide a natural architecture. Approximation properties of those networks are discussed in [12]. Dimensionality reduction can be performed using autoencoders and partial least-squares [13] due to the result by [10,14] (see survey by [15]), and the kernel embedding approach discussed by [16]. Generative models have been considered in Bayesian analysis by [17]. Quantile generative methods proposed in this paper circumvent the need for methods such as MCMC that require density evaluations. Hence, GBC methods provide the practitioner with a useful simulation-based tool for inference and uncertainty quantification in a variety of applied engineering and scientific scenarios.

Connections to Previous Work

Generative models often arise in the context of inverse problems [18,19] and decision-making problems [7]. In the context of inverse problems, the prediction of a mean is rarely an option, since the average of several correct values is not necessarily a correct value and might not even be feasible from the physical point of view. The two main approaches are surrogate-based modeling and approximate Bayes computations (ABC) [15,16,20]. Surrogate-based modeling [21] is a general approach to solving inverse problems, which is based on the availability of a forward model $y = f(\theta)$, which is a deterministic function of parameters θ. The forward model is used to generate a large sample of pairs (y, θ), which is then used to train a surrogate, typically a Gaussian process, which can be used to calculate the inverse map. For a recent review of the surrogate-based approach, see [18]. There are multiple papers that address different aspects of surrogate-based modeling.

Approximate Bayesian Computation (ABC) methods are a generative approach based on local kernel smoothing. There are two major differences to our approach. One, is how the

training dataset is generated. We avoid the use of rejection sampling. The second difference is the use of a quantile neural network (QNN) to approximate the posterior distribution.

In our framework, the map G can be viewed as a nearest neighbor network. ABC relies on the comparison of the summary statistic S calculated from the observed data with the summary statistics calculated from the simulated data. Ref. [22] show that a natural choice of S is via the posterior mean. Ref. [23] shows how to use mixture density networks [24] to approximate the posterior for ABC calculations. In an ABC framework with a parametric exponential family, see [20,25] for the optimal choice of summary statistics. A local smoothing version of ABC is given in [4,26,27], ref. [28] take a basis function approach. Ref. [29] provides an estimation procedure when latent variables are present.

GBC provides an alternative to nonparametric Gaussian process-based surrogates which heavily rely on the informational contribution of each sample point and quickly become ineffective when faced with significant increases in dimensionality [30,31]. Furthermore, homogeneous GP models predict poorly [32]. Unfortunately, the consideration of each input location to handle these heteroskedastic cases results in analytically intractable predictive density and marginal likelihoods [33]. The smoothness assumption made by GP models hinders capturing rapid changes and discontinuities in the input–output relations. Popular attempts to overcome these issues include relying on the selection of kernel functions using prior knowledge about the target process [34]; splitting the input space into subregions so that, inside each of those smaller subregions, the target function is smooth enough and can be approximated with a GP model [35–37]; and learning spatial basis functions [38–40].

For low-dimensional θ, the simplest approach is to discretize the parameter space and the data space and provide a lookup table to approximate $\pi(\theta|y)$. However, this approach is not scalable to high-dimensional θ. For practical cases, when the dimension of θ is high, we can use conditional independence structure present in the data to decompose the joint distribution into a product of lower-dimensional functions [41]. In machine learning, some approaches rely on such a decomposition [41–43]. Most of these approaches use KL divergence as a metric of closeness between the target distribution and the learned distribution. We propose to compute quantiles with the 1-Wasserstein distance as a metric of closeness between the target distribution and the learned distribution.

A natural approach to model the posterior distribution using a neural network is to assume that the parameters of the network are random variables and use MCMC techniques to model the posterior distribution over the parameters [44]. A slightly different approach is to assume that only the weights of the last output layer of a neural network are stochastic [45,46]. GBC provides a natural alternative to these methods.

The rest of the paper is outlined as follows. Section 1 provides a review of the existing literature. Section 2 describes our GBC algorithm. Section 3 describes quantile neural networks (QNNs). Section 4 describes the use of quantile neural networks for Bayesian computations. Section 5 provides two real-world applications from a traffic flow prediction problem and a satellite drag dataset. Finally, Section 6 concludes with directions for future research. There are a number of advantages to our approach. First, it is density-free as we are only estimating quantile functions. Secondly, it can calculate functionals of interest in an efficient manner; see [1].

2. Generative Bayesian Computation (GBC)

Bayesian inference requires samples from the posterior distribution of the parameters given the data. We use the following generic notation.

$$\begin{aligned} y &= \text{ outcome of interest} \\ \theta &= \text{ parameters} \\ \tau &= \text{ base distribution.} \end{aligned}$$

To fix notation, let $\mathcal{Y}$ denote a locally compact metric space of signals, denoted by y, and $\mathcal{B}(\mathcal{Y})$ the Borel σ-algebra of $\mathcal{Y}$. Let λ be a measure on the measurable space of signals $(\mathcal{Y}, \mathcal{B}(\mathcal{Y}))$. Let $P(dy|\theta)$ denote the conditional distribution of signals given the parameters. When a likelihood $p(y|\theta)$ is available w.r.t. the measure λ, we write $P(dy|\theta) = p(y|\theta)\lambda(dy)$.

Let Θ denote a locally compact metric space of admissible parameters (a.k.a. hidden states and latent variables $z \in \mathcal{Z}$) and $\mathcal{B}(\Theta)$ the Borel σ-algebra of Θ. Let μ be a measure on the measurable space of parameters $(\Theta, \mathcal{B}(\Theta))$. Let $\Pi(d\theta|y)$ denote the conditional distribution of the parameters given the observed signal y (a.k.a., the posterior distribution). In many cases, Π is absolutely continuous with density π such that

$$\Pi(d\theta|y) = \pi(\theta|y)\mu(d\theta)$$

We will write $\Pi(d\theta) = \pi(\theta)\mu(d\theta)$ for the prior density π when available.

Specifically, suppose that Y and θ are generated from a model $\mathcal{M}$ and prior to π. This generative approach will allow us to create a training dataset $(\theta^{(i)}, y^{(i)})$ for which we can construct our transport map from a base distribution τ.

Let $Y = y_{\text{obs}}$ denote the observed output. The goal is to be able to draw samples from

$$\theta^{(i)} \sim \pi(\theta|Y = y_{\text{obs}}) \text{ for } 1 \leq i \leq N.$$

Our framework allows for likelihood and density free models. In the case of likelihood-free models, the output is simply specified by a map (a.k.a. forward equation).

Noise Outsourcing Theorem: In many practical applications, the observed ys are high-dimensional, and we can improve the performance of the deep neural network by using a summary statistic $S(y)$. Gere $S : \Re^N \to \Re^k$ is a k-dimensional sufficient statistic, which is a lower-dimensional representation of y. The summary statistic is a sufficient statistic in the Bayes sense [3]. The main idea is to use a deep neural network to approximate the posterior mean $E_\pi(\theta|y)$ as the optimal estimate of $S(y)$.

We propose a non-parametric generative approach to constructing a conditional distribution $P_{Y|X}$ for a given value of the predictor X, we estimate a function $G(\tau, x)$ where τ random variable from the reference distribution, e.g., a uniform, such that $G(\tau, x) \sim P_{Y|X=x}$. To Sample from $P_{Y|X}$, we simply generate τ from the reference distribution and evaluate $G(\tau, x)$. This provides a connection between the conditional distribution estimation and the generalized non-parametric regression. The underpinning is the noise outsourcing theorem ([47], [Theorem 5.10]) which states that any random variable can be represented as a function of another random variable and a noise term. We propose to use a quantile neural network to learn the function $G(\tau, x)$, rather than conditional GANs. Specifically, our goal is to find a function $G : \mathbb{R}^d \times \mathcal{X} \to \mathcal{Y}$ such that the conditional distribution of $G(\tau, X)$, given $X = x$ is the same as the conditional distribution of Y given $X = x$. Since τ is independent of X, it is equivalent of finding a G such that $G(\tau, x) \sim P_{Y|X=x}$, for any $x \in \mathcal{X}$. The existence of G is guaranteed by the noise outsourcing theorem [47], namely,

$$(X, Y) = (X, G(\tau, X)) \text{ a.s., and } G(\tau, x) \sim P_{Y|X=x} \text{ for any } x \in \mathcal{X}.$$

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

$$Y \mid X = x \sim G(\tau, x) \text{ for any } x \in \mathcal{X}.$$

This shows that the problem of finding G is equivalent to a non-linear non-parametric regression.

Now, we apply the noise outsourcing theorem to the problem of calculating a joint distribution Y, θ and conditional distribution $\theta \mid Y$, required for Bayesian inference.

Theorem 1. *If* (Y, θ) *are random variables in a Borel space* $(\mathcal{Y}, \Theta)$ *then there exists an r.v.* $\tau \sim U(0,1)$ *which is independent of* Y *and a function* $G : [0,1] \times \mathcal{Y} \to \Theta$ *such that*

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

Hence the existence of G follows from the noise outsourcing theorem [47]. Moreover, if there is a statistic $S(Y)$ with $Y \perp\!\!\!\perp \theta | S(Y)$, then

$$(\theta | Y = y) \overset{a.s.}{=} G(S(y), \tau).$$

The role of $S(Y)$ is equivalent to the ABC literature. It performs dimension reduction in n, the dimensionality of the signal.

Assuming that we have fitted the deep neural network, G, to the training data, we can use the estimated inverse map to evaluate at new y and τ to obtain a set of posterior samples for any new y by interpolation and evaluating the map

$$\theta \overset{D}{=} \hat{G}_N(S(y), \tau), \quad \text{where} \quad y = (y_1, \ldots, y_n) \tag{1}$$

where $\hat{G}_N$ denotes the estimated map. There are many well-known functional approximation rates for classes of Hölder-smooth functions and deep architectures for G. Ref. [48] provides conditional quantile results, for deep ReLU networks.

A caveat is how to choose G and how well the deep neural network interpolate for the observed input y_{obs}. There is also flexibility in choosing the distribution of τ; for example, τ can also be a high-dimensional vector of Gaussians and essentially provide a mixture-Gaussian approximation for the set of posterior. GBC in a simple way is using pattern matching to provide a look-up table for the map from y to θ. Bayesian computation has then being replaced by the optimization performed by Stochastic Gradient Descent (SGD). Thus, we can sequentially update G using SGD step as new data arrives. MCMC, in comparison, is computationally expensive and needs to be re-run for any new data point. In our examples, we discuss choices of architectures for G and S. Specifically, we propose cosine-embedding for transforming τ.

GBC Algorithm: A necessary condition is the ability to simulate the parameters, latent variables, and data processes. This generates a (potentially large) triple

$$\left\{ y^{(i)}, \theta^{(i)}, \tau^{(i)} \right\}_{i=1}^{N},$$

By construction, the posterior distribution can be characterized by the von Neumann inverse CDF map. For $\theta \in \Re$ we have

$$\theta \overset{D}{=} F^{-1}_{\theta|y}(\tau), \quad \text{where} \quad \tau \sim U(0,1)$$

Given a new base draw τ, we then simply evaluate the following posterior map:

$$\theta \stackrel{D}{=} G(S(y), \tau)$$

When the base draw τ is uniform of same dimension as the parameter vector, the map G is precisely the inverse posterior CDF. This characterizes the posterior distribution $p(\theta|y)$ of interest. As simulation and fitting deep neural networks are computationally cheap, we typically choose N to be of order 10^6 or more.

In many cases, we can estimate a summary statistic, S also via deep learning, and we write the inverse CDF as a composite map, and we have to fit

$$\theta^{(i)} = G(S(y^{(i)}), \tau^{(i)}) \text{ where } F^{-1}_{\theta|y} = G \circ S$$

There are many variations in the architecture design and construction. It is useful to interpret S as a summary statistic that provides a dimension reduction in the y space.

There is flexibility in the choice of base distribution. For example, we can replace τ with a different distribution from which we can easily sample. One example is a multivariate normal, proposed for diffusion models [49]. The dimensionality of the normal can be large. The main insight is that you can solve a high-dimensional least squares problem with non-linearity using stochastic gradient descent. Deep quantile NNs provide a natural candidate for deep learners. Ref. [5] show that implicit quantile networks can be used for distributional reinforcement learning and has proposed the use of quantile regression as a method to minimize 1-Wasserstein in the univariate case when approximating using a mixture of Dirac functions. Figure 1 illustrates the difference in using a quantile objective versus pure density simulation based on the latent variable approach.

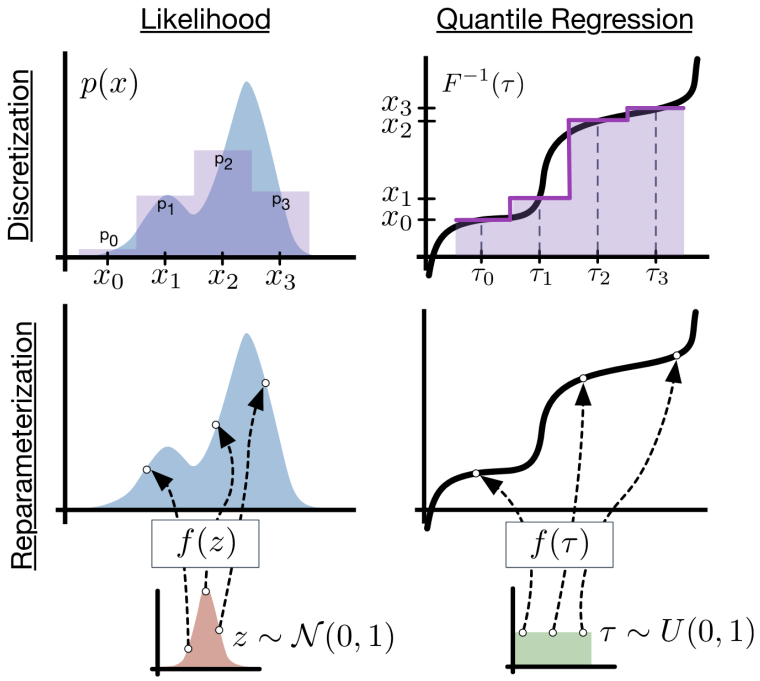


Figure 1. Deep Quantile Neural Network. Source: [6].

Hence, even if y_{obs} is not in the simulated training data set, we can still learn the posterior map of interest. The Kolmogorov–Arnold theorem says that any multivariate function can be expressed this way. So in principle if sample size is large enough we can learn the manifold structure in the parameters for any arbitrary non-linearity. As the dimension of the data y is large, in practice, this requires providing an efficient architecture.

The following Algorithm 1 provides an algorithmic summary of Generative Bayesian Computation (GBC).

Algorithm 1 Generative Bayesian Computation (GBC)

1: Simulate $\theta^{(i)} \sim \pi(\theta)$ and $(y^{(i)} \mid \theta^{(i)})_{1 \le i \le N} \sim p(y \mid \theta)$ or $y^{(i)} = f(\theta^{(i)})$ and $\tau^{(i)} \sim p(\tau)$.
2: Learn a summary statistic $S(y)$ using a deep neural network.
3: Train G using the simulated dataset for $i = 1, \ldots N$, via

$$\theta^{(i)} = G(S(y^{(i)}), \tau^{(i)})$$

4: Given y_{obs}, calculate a sample from $\pi(\theta \mid y)$ with a new base draw $\tau \sim p(\tau)$ by

$$\theta \stackrel{D}{=} \hat{G}_N(\hat{S}_N(y_{\text{obs}}), \tau)$$

using the trained architectures $(\hat{G}_N, \hat{S}_N)$.

In the case when y is high-dimensional, and we can learn a summary statistic $S(y)$ using a deep neural network as an estimate of the posterior mean from the training data, namely using

$$\hat{S}_N(y) = \hat{\theta}(y) = E_\pi(\theta|y).$$

Summary Statistics $S(y)$

Given y, the posterior density is denoted by $\pi(\theta \mid y)$. Here, $y = (y_1, \ldots, y_n)$ is high-dimensional. Moreover, we need the set of posterior probabilities $\pi_{\theta|y}(\theta \in B \mid y)$ for all Borel sets B. Hence, we need dimension reduction for y. The whole idea is to estimate "maps" (a.k.a. transformations/feature extraction) of the output data y; thus, it is reduced to uniformity.

There is a nice connection between the posterior mean and the sufficient statistics, especially the minimal sufficient statistics in the exponential family. If there exists a sufficient statistic S^* for θ, then [3] shows that for almost every y, $\pi(\theta \mid y) = \pi(\theta \mid S^*(y))$, and further $S(y) = E_\pi(\theta \mid y) = E_\pi(\theta \mid S^*(y))$ is a function of $S^*(y)$. In the special case of an exponential family with minimal sufficient statistic S^* and parameter θ, the posterior mean $S(y) = E_\pi(\theta \mid y)$ is a one-to-one function of $S^*(y)$, and thus is a minimal sufficient statistic. Deep learners are good interpolators (folklore theorem of machine learning).

Hence, the set of posteriors $\pi(\theta \mid y)$ is characterized by the distributional identity $\theta \stackrel{D}{=} G(S(y), \tau)$, where $\tau \sim U(0,1)$.

We can construct S using deep learning. Ref. [22] shows that the optimal choice of $S(y) = E_\pi(\theta \mid y)$, namely the posterior mean namely $\hat{\theta}(y)$.

Kolmogorov result on summary statistic: Let $S(y)$ be a sufficient summary statistic in the Bayes sense [3], if for every prior π

$$f_B(y) := \pi_{\theta|y}(\theta \in B \mid y) = \pi_{\theta|s(y)}(\theta \in B \mid s(y)).$$

Then we need to use our pattern matching dataset $(y^{(i)}, \theta^{(i)})$ which is simulated from the prior and forward model to "train" the set of functions $f_B(y)$, where we pick the sets $B = (-\infty, q]$ for a quantile q. Hence, we can then interpolate the mapping to the observed data.

The notion of a summary statistic is prevalent in the ABC literature and is closely related to the notion of a Bayesian sufficient statistic S^* for θ, then [3]), for almost every y,

$$\pi(\theta \mid Y = y) = \pi(\theta \mid S^*(Y) = S^*(y))$$

Furthermore, $S(y) = \mathrm{E}(\theta \mid Y = y) = \mathrm{E}_\pi(\theta \mid S^*(Y) = S^*(y))$ is a function of $S^*(y)$. In the case of exponential family, we have $S(Y) = \mathrm{E}_\pi(\theta|Y)$ is a one-to-one function of $S^*(Y)$, and thus is a minimal sufficient statistic.

Sufficient statistics are generally kept for parametric exponential families, where $S(\cdot)$ is given by the specification of the probabilistic model. However, many forward models have an implicit likelihood and no such structures. The generalization of sufficiency is a summary statistics (a.k.a. feature extraction/selection in a neural network). Hence, we make the assumption that there exists a set of features such that the dimensionality of the problem is reduced.

Learning S can be achieved in a number of ways. First, S is of fixed dimension $S \in \Re^s$ even though $y = (y_1, \ldots y_n)$. Typical architectures include Auto-encoders and traditional dimension reduction methods. Ref. [13] propose to use a theoretical result of Brillinger methods to perform a linear mapping $S(y) = Wy$ and learn W using PLS. Ref. [50] extend this to IV regression and casual inference problems. A key result of [14] shows that we can use linear SGD methods and partial least squares to find $\hat{B}$.

To learn the feature selection variables $S(y)$, ref. [22] propose to use a deep learner to approximate the posterior mean $E_\pi(\theta|y)$ as the optimal estimate of $S(y)$. Specifically, they construct a minimum squared error estimator $\hat{\theta}(y)$ from a large simulated dataset and calculate

$$\min_{\psi} \frac{1}{N} \sum_{i=1}^{N} \|S(y^{(i)}) - \theta^{(i)}\|_2^2$$

where S denotes a DNN. The resulting estimator $\hat{\theta}(y) = \hat{S}_N(y^{(i)})$ approximates the feature vector $S(y)$.

Together with the following architecture for the summary statistic neural network

$$\begin{aligned} S^{(1)} &= ReLU\left(W^{(0)} S^{(0)} + b^{(0)}\right) \\ S^{(2)} &= ReLU\left(W^{(1)} S^{(1)} + b^{(1)}\right) \\ &\vdots \\ S^{(L)} &= ReLU\left(W^{(L-1)} S^{(L-1)} + b^{(L-1)}\right) \\ \hat{\theta} &= W^{(L)} H^{(L)} + b^{(L)}, \end{aligned}$$

where $S^{(0)} = \theta$ is the input, and $\hat{\theta}$ is the summary statistic output.

Double Descent: There is still the question of approximation and the interpolation properties of a DNN. In general, deep learners provide good representations of multivariate functions and are good interpolators. Recent research on interpolation properties of quantile neural networks were recently studied by [51–53]. See also [54,55]. They demonstrate that the generalization error of a DNN exhibits a double descent phenomenon. The authors showed that the test error of the estimator can decrease as the number of parameters increases. The phenomenon of double decent is shown in Figure 2:

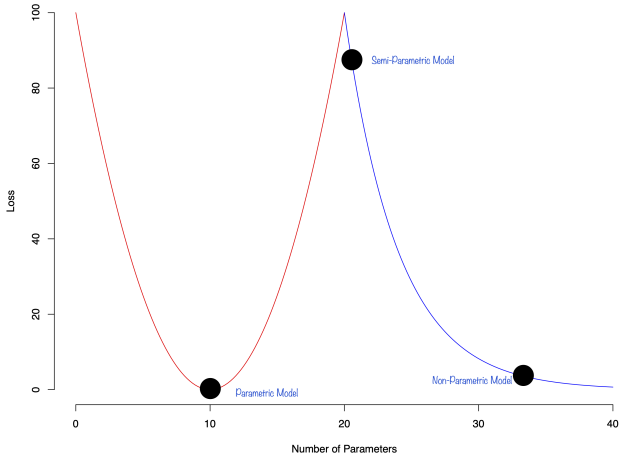


Figure 2. Stylized double descent curve.

The first part of the curve is the classical u-shaped bias-variance trade-off. The second part of the curve is the double descent phenomenon that shows that the test error of the estimator can decrease as the model becomes over-parametrized. This phenomenon was later observed in the context of deep learning [56]. The authors showed that the test error of the estimator can decrease as the number of parameters increases. Ref. [57] discuss theoretical bounds for generalization error for overparametrized models. They show that the generalization error of a model can be bounded by the number of parameters in the model.

The other two sources of error are the training error and the functional approximation error. The training error is a function of N and the functional approximation error depends on the smoothness properties of the underlying posterior and the flexibility of the neural networks used to approximate the posterior.

ABC Approximate Bayesian Computation (ABC) is a generative method for obtaining samples from the posterior distribution. It is a common approach in cases when likelihood is not available, but samples can be generated from some model. The ABC rely on comparing summary statistic calculated from data s_{obs} and of from observed output. We will denote a training sample by input-output pairs $(\theta^{(i)}, Y^{(i)})$. The ABC posterior requires a kernel, a dimensionality reducing summary statistic $S(y)$ and a tolerance level ϵ with a tilted-posterior defined by

$$\pi^{\epsilon}_{ABC}(\theta|Y = y_{\text{obs}}) = \frac{1}{m_{\epsilon}(y_{\text{obs}})} \int K_{\epsilon}(S(y) - y_{\text{obs}})\lambda_{\theta}(dy)\pi(d\theta).$$

Here $y = (y_1, \ldots, y_n)$ is high dimensional. Hence, the need for a k-dimensional summary statistic $S : \Re^n \to \Re^k$ where k is fixed.

This ensures that the mean of the ABC posterior matches that of the posterior of interest. Furthermore, under a uniform kernel $K = I(|S(y) - s_{\text{obs}}| < \epsilon)$ we show convergence

$$\pi^{\epsilon}_{ABC}(\theta \mid Y = y_{\text{obs}}) \to \pi(\theta \mid Y = y_{\text{obs}}).$$

Specifically, the ABC posterior is given by

$$\pi^{\epsilon}_{ABC}(\theta \mid y) = \frac{1}{m_{\epsilon}(y_{\text{obs}})} \int_{\Theta} I(|S(y) - s_{\text{obs}}| < \epsilon)\delta(y - f(\theta))\pi(\theta)dyd\theta = \pi(\theta \mid |S(y) - s_{\text{obs}}| < \epsilon)$$
$$\to \pi(\theta \mid Y = y_{\text{obs}}), \text{ where } S(y) = \hat{\theta}(y) = E_{\pi}(\theta \mid y).$$

A deep NN can effectively learn a good approximation to the posterior mean $E_\pi(\theta|y)$ by using a large data set $(y^{(i)}, \theta^{(i)})_{i=1}^N \sim \pi \times \mathcal{M}$ and solving the ℓ_2 — minimization problem

$$\arg\min_{\psi} \frac{1}{N} \sum_{i=1}^{N} \|S(y^{(i)}) - \theta^{(i)}\|^2.$$

The resulting estimator $\hat{\theta}(y) = \hat{S}(y)$ approaches $S(y)$. This still leaves the problem of simulating a training data set to learn the ABC posterior, which requires the rejection sampling. Our approach, on the other hand, rectifies this issue. Our approach directly evaluates the network on y_{obs} and relies on good generalization error, a.k.a. interpolation.

Conventional ABC methods suffers from the main drawback that the samples do not come from the true posterior, but an approximate one, based on the ϵ-ball approximation of the likelihood, which is a non-parametric local smoother. Theoretically, as ϵ goes to zero, you can guarantee samples from the true posterior. However, the number of sample required is prohibitive. Our method circumvents this by replacing the ball with a deep learning generator and directly model the relationship between the posterior and a baseline uniform Gaussian distribution. Our method is also not a density approximation, as many authors have proposed. Rather, we directly use L_2 methods and Stochastic Gradient Descent to find transport map from θ to a uniform or a Gaussian random variable. The equivalent to the mixture of Gaussian approximation is to assume that our baseline distribution is high-dimensional Gaussian. Such models are called the diffusion models in literature. Full Bayesian computations can then be reduced to high-dimensional L_2 optimization problems with a carefully chosen neural network.

3. Bayes Quantile Neural Networks

One can show that quantile models are direct alternatives to other Bayes computations. Specifically, given $F(y)$, a non-decreasing and continuous from right function. We define

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

which is nondecreasing and continuous from the left.

Now, let $g(y)$ to be a non-decreasing and continuous from left with

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

Then, the transformed quantile has a compositional nature, namely

$$Q_{g(Y)}(u) = g(Q(u))$$

Hence, quantiles act as superposition (a.k.a. deep Learner).

This is best illustrated in the Bayes learning model. We have the following result updating prior to posterior quantiles known as the conditional quantile representation

$$Q_{\theta|Y=y}(u) = Q_Y(v) \;\text{ where }\; v = Q_{F(\theta)|Y=y}(u)$$

To compute v we use

$$u = F_{F(\theta)|Y=y}(v) = P(F(\theta) \leq v|Y = y) = P(\theta \leq Q_\theta(v)|Y = y) = F_{\theta|Y=y}(Q_\theta(v))$$

Ref. [58] also shows the following probabilistic property of quantiles

$$(\theta|Y = y) = Q_{\theta|Y=y}(F_{\theta|y}(\theta)).$$

Hence, we can increase the efficiency by ordering the samples of θ and the baseline distribution, since the mapping being the inverse CDF is monotonic.

3.1. Learning Quantiles

The 1-Wasserstein distance is the ℓ_1 metric of the inverse distribution function. It is also known as earth mover's distance and can be calculated using order statistics [59]. For quantile functions F_U^{-1} and F_V^{-1} the 1-Wasserstein distance is given by

$$W_1(F_U^{-1}, F_V^{-1}) = \int_0^1 |F_U^{-1}(\tau) - F_V^{-1}(\tau)| d\tau.$$

Ref. [60] shows that Wasserstein GANs outperform vanilla GAN due to the improved quantile metric $q = F_U^{-1}(\tau)$ minimize the expected quantile loss

$$E_U[\rho_\tau(u - q)]$$

Quantile regression can be shown to minimize the 1-Wasserstein metric. A related loss is the quantile divergence,

$$q(U, V) = \int_0^1 \int_{F_U^{-1}(q)}^{F_V^{-1}(q)} (F_U(\tau) - q) dq d\tau.$$

The quantile regression likelihood function is an asymmetric function that penalizes overestimation errors with weight τ and underestimation errors with weight $1 - \tau$. For a given input-output pair (x, y), and the quantile function $f(x, \theta)$, parametrized by θ, the quantile loss is $\rho_\tau(u) = u(\tau - I(u < 0))$, where $u = y - f(x)$. From an implementation point of view, a more convenient form of this function is

$$\rho_\tau(u) = \max(u\tau, u(\tau - 1)).$$

Given training data $\{x_i, y_i\}_{i=1}^N$, and a quantile τ, the loss is

$$L_\tau(\theta) = \sum_{i=1}^N \rho_\tau(y_i - f(\tau, x_i, \theta)).$$

Further, we empirically found that adding a means-squared loss to this objective function improves the predictive power of the model, thus the loss function, we use is

$$\alpha L_\tau(\theta) + \frac{1}{N} \sum_{i=1}^N (y_i - f(x_i, \theta))^2.$$

One approach to learn the quantile function is to use a set of quantiles $0 < \tau_1 < \tau_2, \ldots, \tau_K < 1$ and then learn K quantile functions simultaneously by minimizing

$$L(\theta, \tau_1, \ldots, \tau_K) = \frac{1}{NK} \sum_{i=1}^N \sum_{k=1}^K \rho_{\tau_k}(y_i - f_{\tau_k}(x_i, \theta_k)).$$

The corresponding optimization problem of minimizing $L(\theta)$ can be augmented by adding a non-crossing constraint.

$$f_{\tau_i}(x, \theta_i) < f_{\tau_j}(x, \theta_j), \ \forall X, \ i < j.$$

The non-crossing constraint has been considered by several authors, including [61,62].

Cosine Embedding for τ: We will use the following architecture design to learn an inverse CDF (quantile function), namely $F^{-1}(\tau, y) = f_\theta(\tau, y)$, a kernel embedding which augments the input space to the network. The quantile function is represented as a superposition of two other functions $F^{-1}(\tau, y) = f_\theta(\tau, y) = g(\psi(y) \circ \phi(\tau))$ where $\circ$ is the element-wise multiplication operator. Both functions g and ψ are feed-forward neural networks and ϕ is a cosine embedding transform. To avoid overfitting, we use a sufficiently large training dataset; see [5] in a reinforcement learning context. Dimension reduction (a.k.a. feature extraction) will draw on methods used in ABC; see [4] for an approach using semiautomatic approximate to the sufficient statistic.

To learn an inverse CDF (quantile function) $F^{-1}(\tau, y) = f_\theta(\tau, y)$ we will use a kernel embedding trick and augment the predictor space. The quantile function is a superposition for two other functions

$$F^{-1}(\tau, y) = f_\theta(\tau, y) = g(\psi(y) \circ \phi(\tau)),$$

where $\circ$ is the element-wise multiplication operator. Both functions g and ψ are feed-forward neural networks. ϕ is a cosine embedding. To avoid over-fitting, we use a sufficiently large training dataset; see [5] in a reinforcement learning context.

Let g and ψ be feed-forward neural networks and ϕ a cosine embedding given by

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

We now illustrate our approach with a simple synthetic dataset.

3.2. Synthetic Data

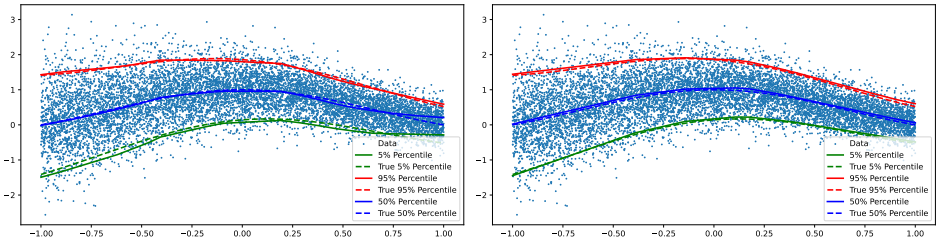
Consider a synthetic data generated from the model

$$x \sim U(-1, 1)$$
$$y \sim N(\sin(\pi x)/(\pi x), \exp(1 - x)/10).$$

The true quantile function is given by

$$f_\tau(x) = \sin(\pi x)/(\pi x) + \Phi^{-1}(\tau)\sqrt{\exp(1 - x)/10},$$

We train two quantile networks, one implicit and one explicit. The explicit network is trained for three fixed quantiles (0.05, 0.5, 0.95). Figure 3 shows fits by both of the networks, we see no empirical difference between the two.



(**a**) (**b**)

Figure 3. We trained both implicit and explicit networks on the synthetic data set. The explicit network was trained for three fixed quantiles (0.05, 0.5, 0.95). We see no empirical difference between the two. (**a**) Implicit quantile network. (**b**) Explicit quantile network.

4. Bayes with Quantiles

Normal–Normal Bayes Learning: Wang Distortion

For the purpose of illustration, we consider the normal–normal learning model. We will develop the necessary quantile theory to show how to calculate posteriors and expected utility without resorting to densities. Also, we show a relationship with Wang's risk distortion measure as the deep learning that needs to be learned.

Specifically, we observe the data $y = (y_1, \ldots, y_n)$ from the following model:

$$\begin{aligned} y \mid \theta &\sim N(\theta, \sigma^2) \\ \theta &\sim N(\mu, \alpha^2) \end{aligned}$$

Hence, the summary (sufficient) statistic $S(y) = \bar{y}$. A remarkable result due to Brillinger, shows that we can learn S independent of G simply via OLS.

Given observed samples $y = (y_1, \ldots, y_n)$, the posterior is then $\theta|y \sim N(\mu_*, \sigma_*^2)$ with

$$\mu_* = (\sigma^2\mu + \alpha^2 s)/t, \quad \sigma_*^2 = \alpha^2\sigma^2/t,$$

where

$$t = \sigma^2 + n\alpha^2 \text{ and } s(y) = \sum_{i=1}^{n} y_i$$

The posterior and prior CDFs are then related via the Wang distortion function

$$1 - \Phi(\theta, \mu_*, \sigma_*) = g(1 - \Phi(\theta, \mu, \alpha^2)),$$

where Φ is the normal distribution function. Here,

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

where

$$\lambda_1 = \frac{\alpha}{\sigma_*} \text{ and } \lambda = \alpha\lambda_1(s - n\mu)/t.$$

The proof is relatively simple and is as follows:

$$\begin{aligned} g(1 - \Phi(\theta, \mu, \alpha^2)) &= g(\Phi(-\theta, \mu, \alpha^2)) = g\left(\Phi\left(-\frac{\theta - \mu}{\alpha}\right)\right) \\ &= \Phi\left(\lambda_1\left(-\frac{\theta - \mu}{\alpha}\right) + \lambda\right) = 1 - \Phi\left(\frac{\theta - (\mu + \alpha\lambda/\lambda_1)}{\alpha/\lambda_1}\right) \end{aligned}$$

with hyperparameters

$$\sigma_* = \alpha/\lambda_1, \quad \lambda_1 = \frac{\alpha}{\sigma_*}$$

and

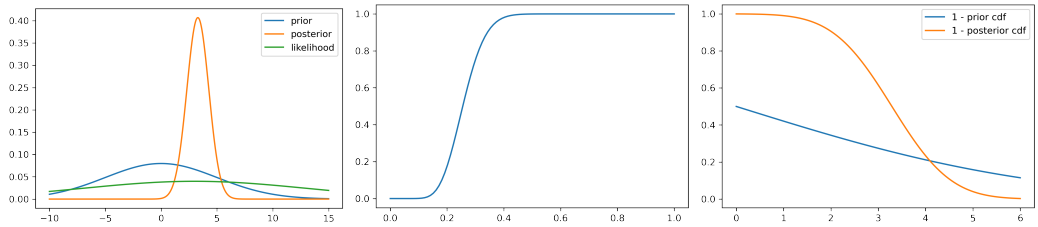
$$\mu_* = \mu + \alpha\lambda/\lambda_1, \quad \lambda = \frac{\lambda_1(\mu_* - \mu)}{\alpha} = \alpha\lambda_1(s - n\mu)/t.$$

We now provide a 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.

The posterior distribution calculated from the sample is then $\theta|y \sim N(3.28, 0.98)$.

Figure 4 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 $1 - \Phi$ for the prior and posterior of the normal–normal model.



(**a**) (**b**) (**c**)

Figure 4. Density for prior, likelihood and posterior, distortion function and 1 - Φ for the prior and posterior of the normal–normal model. (**a**) Model for simulated data. (**b**) Distortion function g. (**c**) $1 - \Phi$.

5. Applications

5.1. Traffic Data

We further illustrate our methodology, using data from a sensor on interstate highway I-55. The sensor is eight miles from downtown Chicago on the northbound I-55 (near Cicero Ave), which is part of a route used by many morning commuters to travel from the suburbs of southwest to the city. As shown in Figure 5, the sensor is located 840 m downstream of an off-ramp and 970 m upstream from an on-ramp.

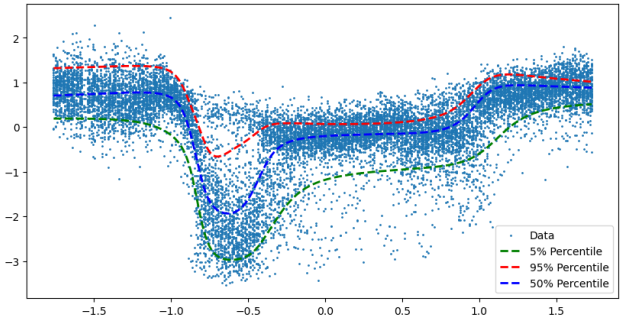


Figure 5. Implicit neural network for traffic speed observed on I-55 north-bound towards Chicago.

In a typical day traffic flow pattern on Chicago's I-55 highway, where sudden breakdowns are followed by a recovery to free flow regime, we can see a breakdown in traffic flow speed during the morning peak period followed by speed recovery. The free flow regimes are usually of little interest to traffic managers. We also see that variance is low during the free-flow regime and high during the breakdown and recovery regimes.

We use the following architecture to model the implicit quantile function.

$$
\begin{aligned}
\tau_1 &= \text{ReLU}\left(w_0^{(1)} + \sum_{i=1}^{64} w_i^{(1)} \cos(i\pi\tau)\right) \\
x_1 &= \text{ReLU}\left(w_0^{(2)} + \sum_{i=1}^{64} w_i^{(2)} x_i\right) \\
z &= \text{ReLU}\left(w_0^{(3)} + \sum_{i=1}^{64} w_i^{(3)} \tau_i x_i\right) \\
y_1 &= \text{ReLU}\left(w_0^{(4)} + \sum_{i=1}^{32} w_i^{(4)} x_i\right) \\
\hat{y} &= w_0^{(5)} + w_i^{(5)} x_i.
\end{aligned}
$$

Our earlier paper [63] analyzed the same data set using sequential Monte Carlo methods and can be used to compare the results.

We now consider a satellite drag example.

5.2. Satellite Drag

Accurate estimation of satellite drag coefficients in low Earth orbit (LEO) is vital for various purposes such as precise positioning (e.g., to plan manoeuvring and determine visibility) and collision avoidance.

Recently, 38 of 49 Starlink satellites launched by SpaceX on 3 February 2022 experienced an early atmospheric reentry caused by unexpectedly elevated atmospheric drag, an estimated $100 MM loss in assets. The launch of the SpaceX Starlink satellites coincided with a geomagnetic storm, which heightened the density of Earth's ionosphere. This, in turn, led to an elevated drag coefficient for the satellites, ultimately causing the majority of the cluster to re-enter the atmosphere and burn up. This recent accident shows the importance of accurate estimation of drag coefficients in commercial and scientific applications [64].

The accurate determination of the drag coefficients is crucial for assessing and maintaining orbital dynamics by taking into account the drag force. Atmospheric drag is the primary source of uncertainty for objects in LEO. This uncertainty arises partially due to inadequate modeling of the interaction between the satellite and the atmosphere. Drag is influenced by various factors, including geometry, orientation, ambient and surface temperatures, and atmospheric chemical composition, all of which are dependent on the position of the satellite (latitude, longitude, and altitude).

Los Alamos National Laboratory developed the Test Particle Monte Carlo simulator to predict the movement of satellites in low earth orbit [65]. The simulator takes two inputs, the geometry of the satellite, given by the mesh approximation and seven parameters, which we list in Table 1 below. The simulator takes about a minute to run one scenario, and we use a dataset of one million scenarios for the Hubble Space Telescope [21]. The simulator outputs estimates of the drag coefficient based on these inputs, while considering uncertainties associated with atmospheric and gas–surface interaction models (GSI).

Table 1. Input parameters for the satellite drag simulator.

Parameter	Range
velocity [m/s]	[5500, 9500]
surface temperature [K]	[100, 500]
atmospheric temperature [K]	[200, 2000]
yaw [radians]	$[-\pi, \pi]$
pitch [radians]	$[-\pi/2, \pi/2]$
normal energy AC [unitless]	[0,1]
tangential momentum AC [unitless]	[0,1]

We use the data set of 1 million simulation runs provided by [2]. The data set has 1 million observations, and we use 20% for training and 80% for testing out-of-sample performance. The model architecture is given below. We use the Adam optimizer and a batch size of 2048, and train the model for 200 epochs.

Ref. [66] provides a survey of modern Gaussian process-based models for prediction and uncertainty quantification tasks. They compare five different models and apply them to the same Hubble data set we use in this section. We use two metrics to assess the quality of the model, namely RMSE, which captures predictive accuracy, and the continuous rank probability score (CRPS; [67,68]). Essentially, CRPS is the absolute difference between the predicted and observed cumulative distribution function (CDF). We used the degenerative

distribution with the entire mass on the observed value (Dirac delta) to obtain the observed CDF. The lower the CRPS, the better.

Their best performing model is treed-GP has the RMSE of 0.08 and CRPS of 0.04, the worst performing model is the deep GP with approximate "doubly stochastic" variational inference that has the RMSE of 0.23 and the CRPS of 0.16. The best performing model in our experiments is the quantile neural network with RMSE of 0.098 and CRPS of 0.05, which is comparable to the top performer from the survey.

Figure 6 plots of the out-of-sample predictions for forty randomly selected responses (green crosses) and compares those to 50th quantile predictions (orange line) and 95% credible prediction intervals (gray region).

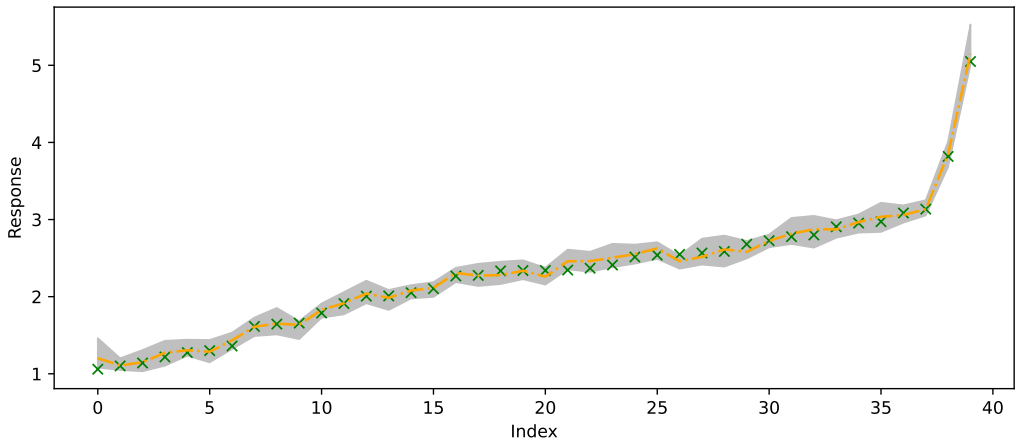
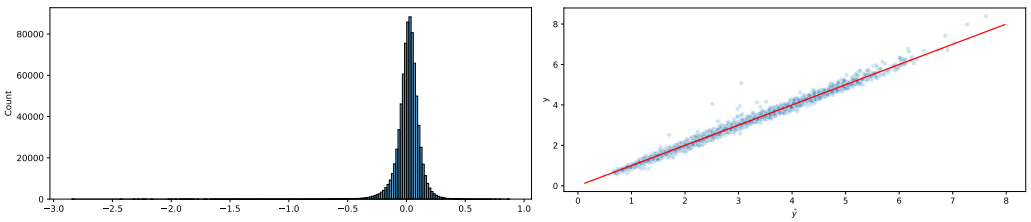


Figure 6. Randomly selected 40 out-of-sample observations from the satellite drag dataset. Green crosses are observed values, orange line is predicted 50% quantile and the gray region is the 95% credible prediction interval.

Figure 7a compares the out-of-sample predictions (50% quantiles) $\hat{y}$ and observed drag coefficients y. We can see that the histogram resembles a normal distribution centered at zero, with some "heaviness" on the left tail, meaning that for some observations, our model underestimates. The scattershot in Figure 7b shows that the model is more accurate for smaller values of y and less accurate for larger values of y and values of y around three.



(**a**) (**b**)

Figure 7. Comparison of out-of-sample predictions (50% quantiles) $\hat{y}$ and observed drag coefficients y. (**a**) Histogram of errors $\hat{y} - y$. (**b**) y vs. $\hat{y}$.

Finally, we show histograms of the posterior predictive distribution for four randomly chosen out-of-sample response values in Figure 8. We can see that the model concentrates the distribution around the true values of the response.

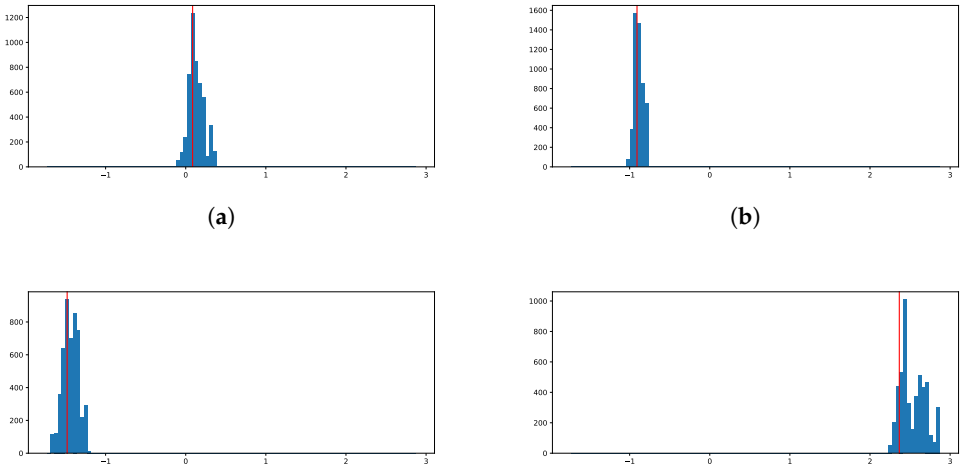


Figure 8. Posterior histograms for four randomly chosen out-of-sample response values. The vertical red line is the observed value of the response. (**a**) Observation 948127. (**b**) Observation 722309. (**c**) Observation 608936. (**d**) Observation 988391.

Overall, our model provides competitive performance to the state-of-the-art techniques used for predicting and uncertainty quantification (UQ) analysis of complex models, such as satellite drag. The model is able to capture the distribution of the response and provide accurate predictions. The model is also capable of providing uncertainty quantification in the form of credible prediction intervals.

6. Discussion

Generative Bayes computations is a simulation-based methodology that takes joint samples of observables and parameters as input and then applies nonparametric regression in a form of deep neural network by regressing θ on a nonlinear function h which is a function of dimensionality-reduced sufficient statistics of θ and a randomly generated stochastically uniform error. In its simplest form, h, can be identified with its inverse CDF.

One solution to the multi-variate case is to use autoregressive quantile neural networks. There are also many alternatives to the architecture design that we propose here. For example, auto-encoder [8,69] or implicit models; see [18,46,70]. There is also a link with indirect inference methods developed in [29,71–73]

There are many challenging future problems. The method can easily handle high-dimensional latent variables. However, designing the architecture for fixed high-dimensional parameters can be challenging, and we will leave this to future research. Having learned the non-linear map, when faced with the observed data y_{obs}, one simply evaluates the non-linear map at newly generated uniform random values. Generative computations circumvent the need for methods like MCMC that require the density evaluations.

We also think that over-parameterization of the problem might lead to a useful model, although it might also lead to nonidentifiability of the weights in the regression. Two interesting approaches include the case where $K > k$ and mixture models with Gaussian mixtures for τ.

Author Contributions: Writing—original draft, N.P. and V.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Polson, N.; Ruggeri, F.; Sokolov, V. Generative Bayesian Computation for Maximum Expected Utility. *Entropy* **2024**, *26*, 1076. [CrossRef] [PubMed]
2. Sun, F.; Gramacy, R.B.; Haaland, B.; Lawrence, E.; Walker, A. Emulating Satellite Drag from Large Simulation Experiments. *SIAM/ASA J. Uncertain. Quantif.* **2019**, *7*, 720–759. [CrossRef]
3. Kolmogorov, A.N. Definition of Center of Dispersion and Measure of Accuracy from a Finite Number of Observations. *Izv. Akad. Nauk SSSR Ser. Mat.* **1942**, *6*, 3–32. (In Russian)
4. Fearnhead, P.; Prangle, D. Constructing Summary Statistics for Approximate Bayesian Computation: Semi-Automatic Approximate Bayesian Computation. *J. R. Stat. Soc. Ser. B (Stat. Methodol.)* **2012**, *74*, 419–474. [CrossRef]
5. Dabney, W.; Ostrovski, G.; Silver, D.; Munos, R. Implicit Quantile Networks for Distributional Reinforcement Learning. *arXiv* **2018**, arXiv:1806.06923. [CrossRef]
6. Ostrovski, G.; Dabney, W.; Munos, R. Autoregressive Quantile Networks for Generative Modeling. *arXiv* **2018**, arXiv:1806.05575. [CrossRef]
7. Dabney, W.; Rowland, M.; Bellemare, M.G.; Munos, R. Distributional Reinforcement Learning with Quantile Regression. *arXiv* **2017**, arXiv:1710.10044. http://arxiv.org/abs/1710.10044. [CrossRef]
8. Albert, C.; Ulzega, S.; Ozdemir, F.; Perez-Cruz, F.; Mira, A. Learning Summary Statistics for Bayesian Inference with Autoencoders. *SciPost Phys. Core* **2022**, *5*, 043. [CrossRef]
9. Müller, P.; West, M.; MacEachern, S.N. Bayesian models for non-linear auto-regressions. *J. Time Ser. Anal.* **1997**, *18*, 593–614. [CrossRef]
10. Bhadra, A.; Datta, J.; Polson, N.; Sokolov, V.; Xu, J. Merging Two Cultures: Deep and Statistical Learning. *arXiv* **2021**, arXiv:2110.11561. http://arxiv.org/abs/2110.11561. [CrossRef]
11. Breiman, L. Statistical Modeling: The Two Cultures (with Comments and a Rejoinder by the Author). *Stat. Sci.* **2001**, *16*, 199–231. [CrossRef]
12. Polson, N.G.; Ročková, V. Posterior Concentration for Sparse Deep Learning. In *Neural Information Processing Systems*; Curran Associates, Inc.: Red Hook, NY, USA, 2018; Volume 31.
13. Polson, N.; Sokolov, V.; Xu, J. Deep Learning Partial Least Squares. *arXiv* **2021**, arXiv:2106.14085. http://arxiv.org/abs/2106.14085.
14. Brillinger, D.R. A Generalized Linear Model With "Gaussian" Regressor Variables. In *Selected Works of David Brillinger*; Guttorp, P., Brillinger, D., Eds.; Selected Works in Probability and Statistics; Springer: New York, NY, USA, 2012; pp. 589–606.
15. Blum, M.G.B.; Nunes, M.A.; Prangle, D.; Sisson, S.A. A Comparative Review of Dimension Reduction Methods in Approximate Bayesian Computation. *Stat. Sci.* **2013**, *28*, 189–208. [CrossRef]
16. Park, M.; Jitkrittum, W.; Sejdinovic, D. K2-ABC: Approximate Bayesian Computation with Kernel Embeddings. In Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, Cadiz, Spain, 9–11 May 2016; pp. 398–407.
17. Zhou, Y.; Gu, Y.; Dunson, D.B. Bayesian Deep Generative Models for Replicated Networks with Multiscale Overlapping Clusters. *arXiv* **2024**, arXiv:2405.20936. http://arxiv.org/abs/2405.20936.
18. Baker, E.; Barbillon, P.; Fadikar, A.; Gramacy, R.B.; Herbei, R.; Higdon, D.; Huang, J.; Johnson, L.R.; Ma, P.; Mondal, A.; et al. Analyzing Stochastic Computer Models: A Review with Opportunities. *Stat. Sci.* **2022**, *37*, 64–89. [CrossRef]
19. Gallant, A.R.; McCulloch, R.E. On the Determination of General Scientific Models with Application to Asset Pricing. *J. Am. Stat. Assoc.* **2009**, *104*, 117–131. [CrossRef]
20. Beaumont, M.A.; Zhang, W.; Balding, D.J. Approximate Bayesian Computation in Population Genetics. *Genetics* **2002**, *162*, 2025–2035. [CrossRef]
21. Gramacy, R.B. *Surrogates: Gaussian Process Modeling, Design, and Optimization for the Applied Sciences*; CRC Press: Boca Raton, FL, USA, 2020.
22. Jiang, B.; Wu, T.Y.; Zheng, C.; Wong, W.H. Learning Summary Statistic For Approximate Bayesian Computation Via Deep Neural Network. *Stat. Sin.* **2017**, *27*, 1595–1618. [CrossRef]
23. Papamakarios, G.; Murray, I. Fast \epsilon -Free Inference of Simulation Models with Bayesian Conditional Density Estimation. In *Advances in Neural Information Processing Systems*; Curran Associates, Inc.: Red Hook, NY, USA, 2016; Volume 29.
24. Bishop, CM. *Mixture Density Networks*; Technical Report NCRG/94/004; Aston University: Birmingham, UK, 1994.
25. Nunes, M.A.; Balding, D.J. On Optimal Selection of Summary Statistics for Approximate Bayesian Computation. *Stat. Appl. Genet. Mol. Biol.* **2010**, *9*. [CrossRef]

26. Jiang, B.; Wu, T.-Y.; Wing H.W. Approximate Bayesian Computation with Kullback-Leibler Divergence as Data Discrepancy. In Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics, Lanzarote, Canary Islands, 9–11 April 2018; pp. 1711–1721.
27. Bernton, E.; Jacob, P.E.; Gerber, M.; Robert, C.P. Approximate Bayesian Computation with the Wasserstein Distance. *J. R. Stat. Soc. Ser. B* **2019**, *81*, 235–269. [CrossRef]
28. Longstaff, F.A.; Schwartz, E.S. Valuing American Options by Simulation: A Simple Least-Squares Approach. *Rev. Financ. Stud.* **2001**, *14*, 113–147. [CrossRef]
29. Pastorello, S.; Patilea, V.; Renault, E. Iterative and Recursive Estimation in Structural Nonadaptive Models. *J. Bus. Econ. Stat.* **2003**, *21*, 449–509. [CrossRef]
30. Shan, S.; Wang, G.G. Survey of modeling and optimization strategies to solve high-dimensional design problems with computationally-expensive black-box functions. *Struct. Multidiscip. Optim.* **2010**, *41*, 219–241. [CrossRef]
31. Donoho, D.L. High-dimensional data analysis: The curses and blessings of dimensionality. In Proceedings of the Ams Conference on Math Challenges of the 21st Century, Lund, Sweden, 6–12 August 2000.
32. Binois, M.; Gramacy, R.B.; Ludkovski, M. Practical heteroskedastic Gaussian process modeling for large simulation experiments. *J. Comput. Graph. Stat.* **2018**, *27*, 808–821. [CrossRef]
33. Lázaro-Gredilla, M.; Quinonero-Candela, J.; Rasmussen, C.E.; Figueiras-Vidal, A.R. Sparse spectrum Gaussian process regression. *J. Mach. Learn. Res.* **2010**, *11*, 1865–1881.
34. Cortes, C.; Haffner, P.; Mohri, M. Rational kernels: Theory and algorithms. *J. Mach. Learn. Res.* **2004**, *5*, 1035–1062.
35. Gramacy, R.B.; Lee, H.K.H. Bayesian Treed Gaussian Process Models with an Application to Computer Modeling. *J. Am. Stat. Assoc.* **2008**, *103*, 1119–1130. [CrossRef]
36. Gramacy, R.B.; Apley, D.W. Local Gaussian Process Approximation for Large Computer Experiments. *J. Comput. Graph. Stat.* **2015**, *24*, 561–578. [CrossRef]
37. Chang, W.; Haran, M.; Olson, R.; Keller, K. Fast dimension-reduced climate model calibration and the effect of data aggregation. *Ann. Appl. Stat.* **2014**, *8*, 649–673. [CrossRef]
38. Bayarri, M.J.; Berger, J.O.; Paulo, R.; Sacks, J.; Cafeo, J.A.; Cavendish, J.; Lin, C.H.; Tu, J. A Framework for Validation of Computer Models. *Technometrics* **2007**, *49*, 138–154. [CrossRef]
39. Wilson, A.G.; Gilboa, E.; Nehorai, A.; Cunningham, J.P. Fast kernel learning for multidimensional pattern extrapolation. *Adv. Neural Inf. Process. Syst.* **2014**, *27*.
40. Higdon, D. Space and space-time modeling using process convolutions. In *Quantitative Methods for Current Environmental Issues*; Springer: Berlin/Heidelberg, Germany, 2002; pp. 37–56.
41. Papamakarios, G.; Pavlakou, T.; Murray, I. Masked Autoregressive Flow for Density Estimation. In *Advances in Neural Information Processing Systems*; Curran Associates, Inc.: Red Hook, NY, USA, 2017; Volume 30.
42. van den Oord, A.; Dieleman, S.; Zen, H.; Simonyan, K.; Vinyals, O.; Graves, A.; Kalchbrenner, N.; Senior, A.; Kavukcuoglu, K. WaveNet: A Generative Model for Raw Audio. *arXiv* **2016**, arXiv:1609.03499. [CrossRef]
43. Germain, M.; Gregor, K.; Murray, I.; Larochelle, H. MADE: Masked Autoencoder for Distribution Estimation. *arXiv* **2015**, arXiv:1502.03509. [CrossRef]
44. Izmailov, P.; Vikram, S.; Hoffman, M.D.; Wilson, A.G.G. What are Bayesian neural network posteriors really like? In Proceedings of the International Conference on Machine Learning, Online, 18–24 July 2021; pp. 4629–4640.
45. Wang, Y.; Polson, N.; Sokolov, V.O. Data Augmentation for Bayesian Deep Learning. *Bayesian Anal.* **2022**, *1*, 1–29. [CrossRef]
46. Schultz, L.; Auld, J.; Sokolov, V. Bayesian Calibration for Activity Based Models. *arXiv* **2022**, arXiv:2203.04414. [CrossRef]
47. Kallenberg, O. *Foundations of Modern Probability*, 2nd ed.; Springer: Berlin/Heidelberg, Germany, 1997.
48. White, H. Some Asymptotic Results for Learning in Single Hidden-Layer Feedforward Network Models. *J. Am. Stat. Assoc.* **1989**, *84*, 1003–1013. [CrossRef]
49. Sohl-Dickstein, J.; Weiss, E.A.; Maheswaranathan, N.; Ganguli, S. Deep Unsupervised Learning Using Nonequilibrium Thermodynamics. *arXiv* **2015**, arXiv:1503.03585. [CrossRef]
50. Nareklishvili, M.; Polson, N.; Sokolov, V. Deep Partial Least Squares for IV Regression. *arXiv* **2022**, arXiv:2207.02612. [CrossRef]
51. Padilla, O.H.M.; Tansey, W.; Chen, Y. Quantile Regression with ReLU Networks: Estimators and Minimax Rates. *J. Mach. Learn. Res.* **2022**, *23*, 247:11251–247:11292.
52. Shen, G.; Jiao, Y.; Lin, Y.; Horowitz, J.L.; Huang, J. Deep Quantile Regression: Mitigating the Curse of Dimensionality Through Composition. *arXiv* **2021**, arXiv:2107.04907. [CrossRef]
53. Schmidt-Hieber, J. Nonparametric Regression Using Deep Neural Networks with ReLU Activation Function. *Ann. Stat.* **2020**, *48*, 1875–1897.
54. Bach, F. High-Dimensional Analysis of Double Descent for Linear Regression with Random Projections. *SIAM J. Math. Data Sci.* **2024**, *6*, 26–50. [CrossRef]

55. Belkin, M.; Rakhlin, A.; Tsybakov, A.B. Does Data Interpolation Contradict Statistical Optimality? In Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics, Naha, Japan, 16–18 April 2019; pp. 1611–1619.
56. Nakkiran, P.; Kaplun, G.; Bansal, Y.; Yang, T.; Barak, B.; Sutskever, I. Deep Double Descent: Where Bigger Models and More Data Hurt. *J. Stat. Mech. Theory Exp.* **2021**, *2021*, 124003. [CrossRef]
57. Cao, Y.; Gu, Q. Generalization Error Bounds of Gradient Descent for Learning Over-Parameterized Deep ReLU Networks. *Proc. AAAI Conf. Artif. Intell.* **2020**, *34*, 3349–3356. [CrossRef]
58. Parzen, E. Quantile Probability and Statistical Data Modeling. *Stat. Sci.* **2004**, *19*, 652–662. [CrossRef]
59. Levina, E.; Bickel, P. The Earth Mover's Distance Is the Mallows Distance: Some Insights from Statistics. In Proceedings of the Proceedings Eighth IEEE International Conference on Computer Vision, ICCV, Vancouver, BC, Canada, 7–14 July 2001; IEEE: Piscataway, NJ, USA, 2001; Volume 2, pp. 251–256.
60. Weng, L. From GAN to WGAN. *arXiv* **2019**, arXiv:1904.08994. [CrossRef]
61. Chernozhukov, V.; Fernández-Val, I.; Galichon, A. Quantile and Probability Curves Without Crossing. *Econometrica* **2010**, *78*, 1093–1125. [CrossRef]
62. Cannon, A.J. Non-Crossing Nonlinear Regression Quantiles by Monotone Composite Quantile Regression Neural Network, with Application to Rainfall Extremes. *Stoch. Environ. Res. Risk Assess.* **2018**, *32*, 3207–3225. [CrossRef]
63. Polson, N.; Sokolov, V. Bayesian Analysis of Traffic Flow on Interstate I-55: The LWR Model. *arXiv* **2014**, arXiv:1409.6034. [CrossRef]
64. Berger, T.E.; Dominique, M.; Lucas, G.; Pilinski, M.; Ray, V.; Sewell, R.; Sutton, E.K.; Thayer, J.P.; Thiemann, E. The Thermosphere Is a Drag: The 2022 Starlink Incident and the Threat of Geomagnetic Storms to Low Earth Orbit Space Operations. *Space Weather* **2023**, *21*, e2022SW003330. [CrossRef]
65. Mehta, P.M.; Walker, A.; Lawrence, E.; Linares, R.; Higdon, D.; Koller, J. Modeling satellite drag coefficients with response surfaces. *Adv. Space Res.* **2014**, *54*, 1590–1607. [CrossRef]
66. Sauer, A.; Cooper, A.; Gramacy, R.B. Non-stationary Gaussian Process Surrogates. *arXiv* **2023**, arXiv:2305.19242. [CrossRef]
67. Gneiting, T.; Raftery, A.E. Strictly proper scoring rules, prediction, and estimation. *J. Am. Stat. Assoc.* **2007**, *102*, 359–378. [CrossRef]
68. Zamo, M.; Naveau, P. Estimation of the continuous ranked probability score with limited information and applications to ensemble weather forecasts. *Math. Geosci.* **2018**, *50*, 209–234. [CrossRef]
69. Akesson, M.; Singh, P.; Wrede, F.; Hellander, A. Convolutional Neural Networks as Summary Statistics for Approximate Bayesian Computation. In *IEEE/ACM Transactions on Computational Biology and Bioinformatics*; IEEE: Piscataway, NJ, USA, 2021.
70. Diggle, P.J.; Gratton, R.J. Monte Carlo Methods of Inference for Implicit Statistical Models. *J. R. Stat. Society. Ser. B (Methodol.)* **1984**, *46*, 193–227. [CrossRef]
71. Stroud, J.R.; Müller, P.; Polson, N.G. Nonlinear state-space models with state-dependent variances. *J. Am. Stat. Assoc.* **2003**, *98*, 377–386. [CrossRef]
72. Drovandi, C.C.; Pettitt, A.N.; Faddy, M.J. Approximate Bayesian Computation Using Indirect Inference. *J. R. Stat. Soc. Ser. C (Appl. Stat.)* **2011**, *60*, 317–337. [CrossRef]
73. Drovandi, C.C.; Pettitt, A.N.; Lee, A. Bayesian Indirect Inference Using a Parametric Auxiliary Model. *Stat. Sci.* **2015**, *30*, 72–95. [CrossRef]