BAYESNDE: BAYESIAN GENERATIVE MODELING FOR NEURAL DENSITY ESTIMATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Density estimation is a fundamental problem in statistics and machine learning. In this work, we introduce BayesNDE, a neural density estimator based on Bayesian generative modeling. BayesNDE learns a Bayesian generative model and evaluates its density without requiring invertible networks or Jacobian-determinant computation. For each observation, it infers a sample-specific latent posterior to construct an adaptive proposal that focuses computation on regions contributing most to its density. Bridge sampling then combines samples from this proposal with separate posterior samples to estimate the density. Experiments on nonlinear and multimodal synthetic datasets show improved estimation of density values and better recovery of the density structure compared to the state-of-the-art neural density estimators. Applications to real-world datasets further demonstrate improved anomaly detection. Together, these results highlight BayesNDE as a flexible and effective neural density estimator, demonstrating how posterior inference can turn generative models into tools for density estimation. The code and tutorials are available at `https://github.com/anonymous`.

1 INTRODUCTION

Density estimation is a fundamental problem in statistics and machine learning. Given observations $\{x_i\}_{i=1}^N$ drawn from an unknown distribution, the goal is to learn the underlying distribution $p(x)$ and evaluate the density value at a new observation. Traditional density estimators such as kernel-based methods (Parzen, 1962) typically perform well only in low-dimensional settings. Density estimation has become increasingly challenging as modern data exhibit complex patterns, such as nonlinearity and high-dimensionality.

Deep neural networks provide powerful tools for modeling complex distributions, offering new advances to density estimation. There are two major types of neural density estimators: autoregressive models and normalizing flows. Autoregressive models obtain tractable likelihoods through a sequential factorization of the joint density (Germain et al., 2015; Papamakarios et al., 2017), while normalizing flows construct invertible transformations whose Jacobian determinants can be efficiently evaluated (Dinh et al., 2017; Chen et al., 2018; 2019).

Despite their different formulations, both autoregressive density estimators and normalizing flows can be viewed within a general change-of-variables framework (Kingma et al., 2016). Specifically, let $z \sim p_Z(z)$ and $x = G(z)$, where $G : \mathbb{R}^p \to \mathbb{R}^p$ is differentiable and invertible. The density of x follows,

$$p_X(x) = p_Z(z) \left| \det \frac{\partial G(z)}{\partial z^\top} \right|^{-1}. \tag{1}$$

Within this change-of-variables formulation, exact density evaluation requires both invertible transformation and tractable Jacobian computation. It requires the latent and observed spaces to have the same dimension and constrains the neural architecture, which may potentially limit the expressiveness of the neural architecture.

Latent-variable generative models provide an alternative route to flexible distribution modeling. Given a latent variable $\boldsymbol{z}$ with prior $p_Z(\boldsymbol{z})$ and the model $p_{\boldsymbol{\theta}}(\boldsymbol{x} \mid \boldsymbol{z})$, the marginal density is

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \int p_{\boldsymbol{\theta}}(\boldsymbol{x} \mid \boldsymbol{z}) p_Z(\boldsymbol{z})\, d\boldsymbol{z}. \tag{2}$$

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

The challenge is therefore shifted from Jacobian computation to integration over the latent space, which is generally intractable. Following this direction, Liu et al. (2021) proposed Roundtrip, which models observations around a learned low-dimensional manifold and evaluates pointwise densities using importance sampling or Laplacian approximation.

Although Roundtrip substantially relaxes the architectural constraints of flow-based density estimators, its importance proposal is centered at the encoder output, which provides a deterministic latent representation rather than a distributional characterization of the observation-specific posterior. The resulting proposal may therefore not fully capture posterior uncertainty, dependence, or multimodality, potentially reducing importance-sampling efficiency when proposal-posterior mismatch is substantial.

To overcome these limitations, we introduce BayesNDE, a neural density estimator based on Bayesian generative modeling. To our knowledge, BayesNDE is the first neural density-estimation framework to use sample-specific posterior-adaptive bridge sampling for explicit pointwise density evaluation. We summarize our contributions as follows:

- We introduce a new perspective for explicit neural density evaluation by estimating the marginal or conditional density of each observation through posterior-adaptive bridge sampling. This removes the need for dimension-preserving, invertible transformations and tractable Jacobian determinants.
- BayesNDE uses observation-specific posterior samples to construct adaptive proposals and bridge sampling to estimate the resulting normalizing constants. We establish consistency of the estimator and characterize how posterior-proposal overlap determines its asymptotic efficiency.
- BayesNDE achieves superior performance in a diverse of density estimation benchmarks, including nonlinear and multimodal simulation datasets, and real-world datasets. The downstream applications, such as outlier detection and Bayesian classification, further support the significance of our method.

2 Related Work

Neural density estimation. Modern neural density estimators are largely built around model classes that admit tractable likelihood evaluation. Autoregressive models, such as MADE (Germain et al., 2015), represent a joint density through sequential conditional factorizations, while normalizing flows construct invertible transformations with tractable change-of-variables computation, including MAF (Papamakarios et al., 2017), Real NVP (Dinh et al., 2017), and Residual Flow (Resflow) (Chen et al., 2019). Continuous normalizing flows extend this framework to continuous-time transformations (Chen et al., 2018), while Flow Matching provides an efficient approach for learning the associated vector fields (Lipman et al., 2023). Relatedly, Wu et al. (2017) used annealed importance sampling to evaluate test log-likelihoods of pretrained decoder-based generative models. Their focus is quantitative evaluation of fitted generative models rather than the development and benchmarking of a general-purpose neural density estimator.

Normalizing-constant estimation. Estimating normalizing constants is a classical problem in Bayesian computation. Importance sampling estimates an unknown normalizing constant using weighted samples from a proposal distribution, with efficiency depending critically on proposal-target overlap. Bridge sampling estimates the normalizing constant by combining samples from both a proposal distribution and the normalized target distribution (Meng & Wong, 1996; Gronau et al., 2020). Related adaptive importance-sampling methods further use information from posterior regions to construct more effective proposal distributions (Hoogerheide et al., 2012). (Jia & Seljak, 2020) adopt bridge sampling with a normalizing-flow-based proposal for normalizing-constant estimation of Bayesian evidence. These methods, however, are designed for generic normalizing-constant estimation rather than neural density estimation of the observed-data distribution.

Positioning of BayesNDE. BayesNDE bridges the gap between neural density estimation and Bayesian normalizing-constant estimation through observation-specific posterior inference. Its generative model is inspired by recent advances in Bayesian generative modeling (Liu & Wong,

2026b;a; Luo & Liu, 2026), while our focus is on explicit density estimation through posterior-adaptive bridge sampling. BayesNDE exploits this posterior to guide latent-space integration, providing a new perspective on neural density estimation.

3 METHOD

3.1 METHOD OVERVIEW

BayesNDE proceeds in two stages (Figure 1). First, we train a Bayesian generative model to capture the underlying data distribution. Second, with the fitted model, we estimate the density of a new observation as the normalizing constant of its latent posterior. Posterior samples guide the construction of an observation-adaptive proposal that concentrates computation in latent regions contributing most to the marginal-density integral. Bridge sampling then combines proposal draws with held-out posterior samples to estimate this normalizing constant and obtain the pointwise density.

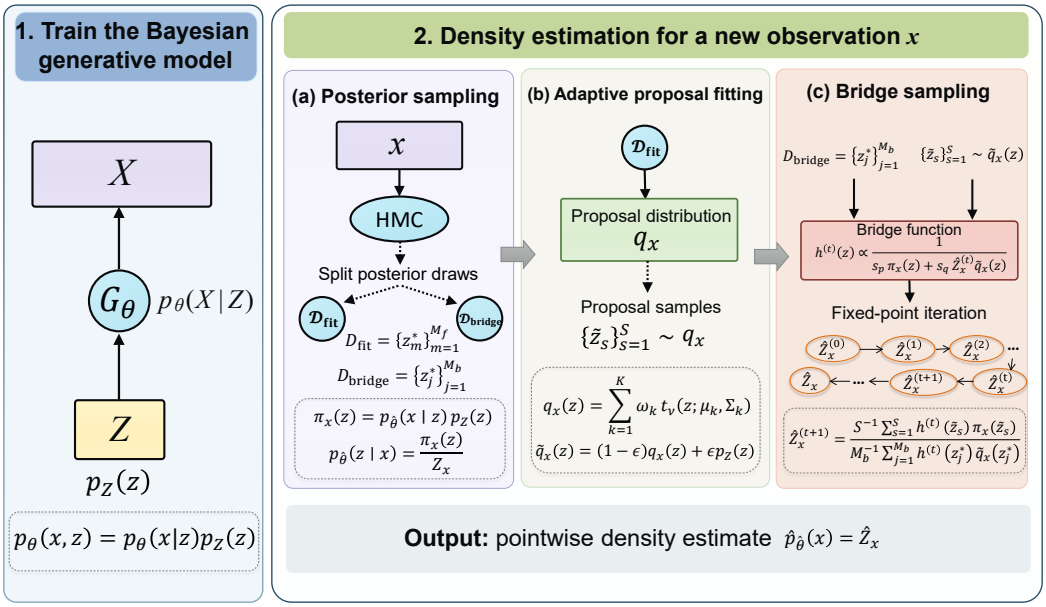


Figure 1: Overview of BayesNDE.

Specifically, for a fitted parameter $\hat{\boldsymbol{\theta}}$ and an observation $\boldsymbol{x}$, define the unnormalized latent posterior as $\pi_{\boldsymbol{x}}(\boldsymbol{z}) = p_{\hat{\boldsymbol{\theta}}}(\boldsymbol{x} \mid \boldsymbol{z}) p_Z(\boldsymbol{z})$. Its normalizing constant is exactly the desired marginal density,

$$Z_{\boldsymbol{x}} := \int \pi_{\boldsymbol{x}}(\boldsymbol{z})\, d\boldsymbol{z} = p_{\hat{\boldsymbol{\theta}}}(\boldsymbol{x}), \qquad p_{\hat{\boldsymbol{\theta}}}(\boldsymbol{z} \mid \boldsymbol{x}) = \frac{\pi_{\boldsymbol{x}}(\boldsymbol{z})}{Z_{\boldsymbol{x}}}. \tag{3}$$

Thus, pointwise density evaluation reduces to estimating $Z_{\boldsymbol{x}}$. Posterior inference requires only the unnormalized posterior $\pi_{\boldsymbol{x}}(\boldsymbol{z})$, allowing posterior samples to identify the latent regions that contribute most to the marginal-density integral. BayesNDE uses these samples to construct an observation-specific proposal and estimates $Z_{\boldsymbol{x}}$ by bridge sampling.

3.2 BAYESIAN GENERATIVE MODEL

Let $\mathbf{x} \in \mathbb{R}^p$ denote the observed random vector and $\mathbf{z} \in \mathbb{R}^d$ a low-dimensional latent random vector, where typically $d < p$. We assume a standard Gaussian prior $p_Z(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}; \mathbf{0}, \boldsymbol{I}_d)$ for the latent variable. For continuous data, the conditional distribution of $\mathbf{x}$ given $\mathbf{z}$ is modeled as

$$p_{\boldsymbol{\theta}}(\boldsymbol{x} \mid \boldsymbol{z}) = \mathcal{N}\left(\boldsymbol{x}; \boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{z}), \boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\boldsymbol{z})\right), \tag{4}$$

where $\boldsymbol{\theta}$ denotes the parameters of the generative model. Both the conditional mean $\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{z})$ and covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\boldsymbol{z})$ are represented by a neural network with two output heads. For simplicity,

we use the diagonal structure $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\boldsymbol{z}) = \operatorname{diag}\left(\sigma_1^2(\boldsymbol{z};\boldsymbol{\theta}), \ldots, \sigma_p^2(\boldsymbol{z};\boldsymbol{\theta})\right)$. Importantly, the Gaussian assumption is imposed conditionally on $\mathbf{z}$; a nonlinear marginal distribution of $\boldsymbol{x}$ is induced after integrating over the latent variable.

During model fitting, we first use an EGM warm start (Liu et al., 2024; Liu & Wong, 2026b) to initialize the generative parameters and sample-specific latent representations (see details in Appendix A.4). Then training proceeds by iteratively updating the sample-specific latent variables $\{\boldsymbol{z}_i\}_{i=1}^N$ and the shared generative parameters $\boldsymbol{\theta}$. Given the current $\boldsymbol{\theta}$, the latent variable associated with observation $\boldsymbol{x}_i$ is updated according to its log-posterior $\log p_{\boldsymbol{\theta}}(\boldsymbol{z}_i \mid \boldsymbol{x}_i) = \log p_Z(\boldsymbol{z}_i) + \log p_{\boldsymbol{\theta}}(\boldsymbol{x}_i \mid \boldsymbol{z}_i) + C_i$, where C_i does not depend on $\boldsymbol{z}_i$. Since the right-hand side is differentiable with respect to $\boldsymbol{z}_i$, each latent variable is updated by stochastic gradient ascent. These updates are fully decoupled across observations and can therefore be performed in parallel to improve efficiency.

Conditional on the current latent variables, the generative parameters are updated using the mini-batch conditional log-likelihood $l_{\mathcal{B}}(\boldsymbol{\theta}) = \frac{1}{|\mathcal{B}|}\sum_{i\in\mathcal{B}} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}_i \mid \boldsymbol{z}_i)$, together with the regularization terms specified in Appendix A.4. The two updates are alternated until convergence. The fitted generative parameters $\hat{\boldsymbol{\theta}}$ are then fixed for subsequent pointwise density evaluation.

3.3 Posterior Sampling and Proposal Construction

For a new observation $\boldsymbol{x}$, the fitted model defines the latent posterior

$$p_{\hat{\boldsymbol{\theta}}}(\boldsymbol{z} \mid \boldsymbol{x}) = \frac{\pi_{\boldsymbol{x}}(\boldsymbol{z})}{Z_{\boldsymbol{x}}} = \frac{p_{\hat{\boldsymbol{\theta}}}(\boldsymbol{x} \mid \boldsymbol{z}) p_Z(\boldsymbol{z})}{p_{\hat{\boldsymbol{\theta}}}(\boldsymbol{x})}. \tag{5}$$

Although the normalizing constant $Z_{\boldsymbol{x}}$ is unknown, posterior sampling only requires evaluation of $\pi_{\boldsymbol{x}}(\boldsymbol{z})$. We therefore use Hamiltonian Monte Carlo (HMC) (Duane et al., 1987; Neal, 2011) to obtain posterior draws $\{\boldsymbol{z}_m^\star\}_{m=1}^M$ from $p_{\hat{\boldsymbol{\theta}}}(\boldsymbol{z} \mid \boldsymbol{x})$. To construct a proposal, the posterior draws are randomly split into two disjoint sets, $\mathcal{D}_{\text{fit}}$ and $\mathcal{D}_{\text{bridge}}$, following the sample-splitting principle studied for bridge sampling (Wong et al., 2020). The former is used exclusively for proposal construction, while the latter is retained for subsequent bridge sampling.

To capture the observation-specific posterior geometry, we first fit a K-component full-covariance Gaussian mixture model (GMM) to $\mathcal{D}_{\text{fit}}$ and use the fitted mixture weights, means, and regularized covariance matrices to construct a mixture-of-Student-t proposal $q_{\boldsymbol{x}}(\boldsymbol{z}) = \sum_{k=1}^K \omega_k t_\nu\left(\boldsymbol{z}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)$, where ν denotes the degrees of freedom. The mixture structure accommodates multimodal posterior geometry, while the heavy-tailed components improve coverage of the posterior tails.

To improve robustness to proposal misspecification and provide additional tail coverage, we further introduce a defensive component based on the latent prior (Hesterberg, 1995; Owen & Zhou, 2000), $\widetilde{q}_{\boldsymbol{x}}(\boldsymbol{z}) = (1-\epsilon) q_{\boldsymbol{x}}(\boldsymbol{z}) + \epsilon p_Z(\boldsymbol{z})$, where $\epsilon \in (0,1)$ controls the defensive weight. We then draw S independent proposal samples $\{\widetilde{\boldsymbol{z}}_s\}_{s=1}^S$ from $\widetilde{q}_{\boldsymbol{x}}$. Together with the held-out posterior draws in $\mathcal{D}_{\text{bridge}}$, these samples are used for pointwise density estimation in the following section.

3.4 Bridge-Sampling Density Estimation

We estimate the normalizing constant $Z_{\boldsymbol{x}} = p_{\hat{\boldsymbol{\theta}}}(\boldsymbol{x})$ using bridge sampling. Using the defensive proposal $\widetilde{q}_{\boldsymbol{x}}$, we first obtain an importance-sampling estimate

$$\widehat{Z}_{\text{IS}} = \frac{1}{S}\sum_{s=1}^S \frac{\pi_{\boldsymbol{x}}(\widetilde{\boldsymbol{z}}_s)}{\widetilde{q}_{\boldsymbol{x}}(\widetilde{\boldsymbol{z}}_s)}, \qquad \widetilde{\boldsymbol{z}}_s \sim \widetilde{q}_{\boldsymbol{x}}. \tag{6}$$

This estimate is used to initialize the bridge-sampling iteration. For any positive bridge function h, the bridge identity and its Monte Carlo approximation are

$$Z_{\boldsymbol{x}} = \frac{\mathbb{E}_{\widetilde{q}_{\boldsymbol{x}}}\left[h(\boldsymbol{z})\pi_{\boldsymbol{x}}(\boldsymbol{z})\right]}{\mathbb{E}_{p_{\hat{\boldsymbol{\theta}}}(\boldsymbol{z}\mid\boldsymbol{x})}\left[h(\boldsymbol{z})\widetilde{q}_{\boldsymbol{x}}(\boldsymbol{z})\right]}, \qquad \widehat{Z}_{\boldsymbol{x}} = \frac{S^{-1}\sum_{s=1}^S h(\widetilde{\boldsymbol{z}}_s)\pi_{\boldsymbol{x}}(\widetilde{\boldsymbol{z}}_s)}{M_{\text{b}}^{-1}\sum_{j=1}^{M_{\text{b}}} h(\boldsymbol{z}_j^\star)\widetilde{q}_{\boldsymbol{x}}(\boldsymbol{z}_j^\star)}. \tag{7}$$

BayesNDE uses the asymptotically optimal bridge function (Meng & Wong, 1996)

$$h(\boldsymbol{z}) \propto \frac{1}{s_p \pi_{\boldsymbol{x}}(\boldsymbol{z}) + s_q Z_{\boldsymbol{x}} \widetilde{q}_{\boldsymbol{x}}(\boldsymbol{z})}, \qquad s_p = \frac{M_{\text{eff}}}{M_{\text{eff}} + S}, \quad s_q = \frac{S}{M_{\text{eff}} + S}, \tag{8}$$

where M_{eff} denotes an effective posterior sample count computed from the held-out bridge draws, as detailed in Appendix A.4. Because $h^\star(\boldsymbol{z})$ itself depends on the unknown normalizing constant, $Z_{\boldsymbol{x}}$ is obtained through the fixed-point update

$$\widehat{Z}_{\boldsymbol{x}}^{(t+1)} = \frac{\dfrac{1}{S}\displaystyle\sum_{s=1}^{S} \dfrac{\pi_{\boldsymbol{x}}(\widetilde{\boldsymbol{z}}_s)}{s_p \pi_{\boldsymbol{x}}(\widetilde{\boldsymbol{z}}_s) + s_q \widehat{Z}_{\boldsymbol{x}}^{(t)} \widetilde{q}_{\boldsymbol{x}}(\widetilde{\boldsymbol{z}}_s)}}{\dfrac{1}{M_{\text{b}}}\displaystyle\sum_{j=1}^{M_{\text{b}}} \dfrac{\widetilde{q}_{\boldsymbol{x}}(\boldsymbol{z}_j^\star)}{s_p \pi_{\boldsymbol{x}}(\boldsymbol{z}_j^\star) + s_q \widehat{Z}_{\boldsymbol{x}}^{(t)} \widetilde{q}_{\boldsymbol{x}}(\boldsymbol{z}_j^\star)}}. \tag{9}$$

The iteration is initialized at $\widehat{Z}_{\boldsymbol{x}}^{(0)} = \widehat{Z}_{\text{IS}}$, and continued until convergence. The resulting $\log \widehat{p}_{\hat{\boldsymbol{\theta}}}(\boldsymbol{x}) = \log \widehat{Z}_{\boldsymbol{x}}$ is reported as the estimated pointwise log-density.

For conditional density estimation with an observed conditioning variable $\boldsymbol{y}$, we extend the Bayesian generative model to a conditional generative architecture and the same procedure applies after replacing $p_{\hat{\boldsymbol{\theta}}}(\boldsymbol{x} \mid \boldsymbol{z})$ by $p_{\hat{\boldsymbol{\theta}}}(\boldsymbol{x} \mid \boldsymbol{z}, \boldsymbol{y})$ throughout, yielding an estimate of $p_{\hat{\boldsymbol{\theta}}}(\boldsymbol{x} \mid \boldsymbol{y})$. See details in Appendix A.3.

3.5 Training Objectives and Model Architecture

We briefly summarize the training objectives and network architectures used to instantiate the Bayesian generative model. For EGM initialization, the generator G and encoder E are trained using a weighted combination of adversarial matching, bidirectional reconstruction, distribution matching through MMD (Gretton et al., 2012) and sliced-Wasserstein discrepancy (Kolouri et al., 2019), correlation matching, and conditional-variance regularization:

$$\begin{aligned}\mathcal{L}_{\text{EGM}} = \lambda_{\text{adv}}\mathcal{L}_{\text{adv}} + \lambda_{\text{rec}}\mathcal{L}_{\text{rec}} + \lambda_{\text{corr}}\mathcal{L}_{\text{corr}} \\ + \lambda_{\text{MMD}}\mathcal{L}_{\text{MMD}} + \lambda_{\text{SW}}\mathcal{L}_{\text{SW}} + \lambda_{\text{var}}\mathcal{L}_{\text{logvar}},\end{aligned} \tag{10}$$

where the adversarial and reconstruction terms encourage agreement between the data and latent representations, while the remaining terms regularize distributional, dependence, and conditional-variance properties. The detailed experimental settings are reported in Appendix A.4.

During the subsequent stochastic iterative updating, the generative parameters are optimized using the conditional negative log-likelihood together with the regularization terms:

$$\mathcal{J}_{\text{iter}}(\boldsymbol{\theta}) = -l_{\mathbb{B}}(\boldsymbol{\theta}) + \lambda_{\text{PP}}\mathcal{L}_{\text{MMD,PP}} + \lambda_{\text{corr}}^{\text{iter}}\mathcal{L}_{\text{corr}} + \lambda_{\text{var}}^{\text{iter}}\mathcal{L}_{\text{logvar}}, \tag{11}$$

where $\mathcal{L}_{\text{MMD,PP}}$ matches the prior-predictive distribution to the observed data.

For vector-valued data, the generative model $G_{\boldsymbol{\theta}}$ adopts five width-256 hidden layers for $p \leq 10$ and five width-256 residual blocks for $p > 10$ with separate conditional-mean and diagonal-variance heads, with a Softplus transformation to ensure positive variances. Detailed loss definitions, architecture settings, optimization hyperparameters, and regime-specific weights are provided in Appendix A.4.

4 Theory

4.1 Consistency of Adaptive Bridge Estimation

BayesNDE constructs a bridge proposal from posterior samples, so the proposal itself is data-adaptive. We show that this adaptation does not alter the asymptotic target of the density estimator, provided that the proposal-fitting samples are separated from those used in bridge estimation.

Theorem 1 (Consistency of posterior-adaptive bridge estimation)**.** *Fix an observation x and a fitted generative model $\hat{\boldsymbol{\theta}}$. Let $\pi_x(z) = p_{\hat{\boldsymbol{\theta}}}(x \mid z)p_Z(z)$ and $Z_x = \int \pi_x(z)\,dz = p_{\hat{\boldsymbol{\theta}}}(x)$. Suppose that the adaptive proposal is constructed from a separate posterior sample, has support wherever π_x is positive, and that the proposal and held-out posterior samples satisfy the corresponding laws of large numbers. Then the bridge estimate $\widehat{Z}_x$ produced by BayesNDE satisfies*

$$\widehat{Z}_x \xrightarrow{p} Z_x = p_{\hat{\boldsymbol{\theta}}}(x)$$

as the numbers of proposal and posterior samples tend to infinity.

Theorem 1 shows that posterior adaptation affects the efficiency of bridge estimation, but not its asymptotic target. The result is conditional on the fitted generative model and therefore concerns the consistency of the Monte Carlo density evaluation step, rather than consistency with respect to the unknown data-generating density. The full assumptions and proof under the regularity conditions are given in Appendix A.1.

4.2 Posterior–proposal overlap and estimation efficiency

Classical bridge-sampling theory shows that the bridge function and the overlap between the two sampling distributions determine the efficiency of normalizing-constant estimation (Meng & Wong, 1996; Gronau et al., 2020). We specialize this analysis to the latent posterior $p_{\boldsymbol{x}} = \pi_{\boldsymbol{x}}/Z_{\boldsymbol{x}}$ and the defensive proposal $\tilde{q}_{\boldsymbol{x}}$, treated as fixed given $\mathcal{D}_{\text{fit}}$. Let $N = M_b + S$ with $M_b/N \to s_p \in (0,1)$ and $s_q = 1 - s_p$, and define, for any nonnegative function q,

$$\mathcal{O}_{\boldsymbol{x}}(q) = \int \frac{p_{\boldsymbol{x}}(\boldsymbol{z})\, q(\boldsymbol{z})}{s_p\, p_{\boldsymbol{x}}(\boldsymbol{z}) + s_q\, q(\boldsymbol{z})}\, \mathrm{d}\boldsymbol{z}. \tag{12}$$

Theorem 2 (Efficiency under posterior–proposal overlap)**.** *Suppose the M_b held-out draws are i.i.d. from $p_{\boldsymbol{x}}$ and the S proposal draws are i.i.d. from $\tilde{q}_{\boldsymbol{x}}$, independently of each other. For the optimal bridge estimator of Meng & Wong (1996),*

$$\sqrt{N}\big(\widehat{Z}_{\boldsymbol{x}}/Z_{\boldsymbol{x}} - 1\big) \xrightarrow{d} \mathcal{N}\Big(0,\ \tfrac{1}{s_p s_q}\big\{\mathcal{O}_{\boldsymbol{x}}(\tilde{q}_{\boldsymbol{x}})^{-1} - 1\big\}\Big).$$

Moreover, $0 < \mathcal{O}_{\boldsymbol{x}}(\epsilon\, p_Z) \le \mathcal{O}_{\boldsymbol{x}}(\tilde{q}_{\boldsymbol{x}}) \le 1$, where the lower bound does not depend on the fitted mixture $q_{\boldsymbol{x}}$ and the upper bound is attained if and only if $\tilde{q}_{\boldsymbol{x}} = p_{\boldsymbol{x}}$ almost everywhere.

No overlap or moment conditions are required, since the bridge integrands are bounded by $1/s_p$ and $1/s_q$. Stronger overlap between $\tilde{q}_{\boldsymbol{x}}$ and $p_{\boldsymbol{x}}$ yields smaller Monte Carlo variance, which motivates observation-specific proposals, while the defensive component bounds the variance however poorly $q_{\boldsymbol{x}}$ fits the posterior. For correlated HMC draws, replacing M_b by M_{eff} in (s_p, s_q) is a heuristic. The proof is given in Appendix A.1.

5 Results

5.1 Experimental setup

Datasets and evaluation. We evaluate BayesNDE on both synthetic benchmarks with known densities and real-world benchmarks. The synthetic experiments are designed to reflect two complementary challenges in density estimation. The independent Gaussian mixture (Independent GMM) represents a strongly multimodal distribution: each coordinate independently follows an equally weighted three-component Gaussian mixture with means -1, 0, and 1, yielding 3^p modes in p dimensions. The involute distribution instead represents a highly nonlinear density concentrated around a curved manifold. Detailed data generative processes are provided in Appendix A.2. For each distribution, we generate $20{,}000$ i.i.d. observations and use $81\%/9\%/10\%$ for training, validation, and testing.

In two dimensions, estimated densities are compared directly with the ground truth on a common grid; for the independent Gaussian mixture, we further evaluate dimensions from 2 to 25 using the Spearman rank correlation between true and estimated test densities. For real data, we use five tabular benchmarks from the UCI Machine Learning Repository (Dua & Graff, 2017). BANK and ParkTele are used for unconditional density estimation, while Pendigits10, Vehicle, and EEGEye are used for conditional density estimation with one-hot encoded class labels. Performance is measured by average test log-likelihood.

Baseline methods. We compare BayesNDE with six leading neural density estimators spanning autoregressive models, normalizing flows, continuous flows, and generative latent-variable models: MADE (Germain et al., 2015), MAF (Papamakarios et al., 2017), Real NVP (Dinh et al., 2017), Residual Flow (Resflow) (Chen et al., 2019), Conditional Flow Matching (CFMs) (Lipman et al., 2023), and Roundtrip (Liu et al., 2021). These baselines cover widely used approaches based on autoregressive factorization, invertible transformations, continuous-time flows, and flexible deep generative mappings. Implementation details of the baselines are provided in Appendix A.5.

5.2 SIMULATION STUDIES

We first evaluate whether the estimated densities recover the underlying structure of the two synthetic distributions. Figure 2 shows density maps of the true and estimated densities in two dimensions. BayesNDE closely recovers both the nonlinear geometry of the involute distribution and the nine separated modes of the Gaussian mixture, while competing methods exhibit varying degrees of smoothing, distortion, or spurious connections between modes. For example, on the involute benchmark, BayesNDE achieves a Spearman correlation of 0.959, compared with 0.898 for the strongest baseline, Roundtrip.

We next examine how density estimation accuracy changes as the dimension increases. Using the Gaussian mixture benchmark, each coordinate independently follows a three-component mixture, the number of modes grows exponentially as 3^p, making the problem increasingly challenging with dimension. As shown in Figure 4a, BayesNDE achieves the highest Spearman correlation at every evaluated dimension. For example, when $p = 20$, BayesNDE attains a correlation of 0.57, compared with 0.43 for the strongest competing method.

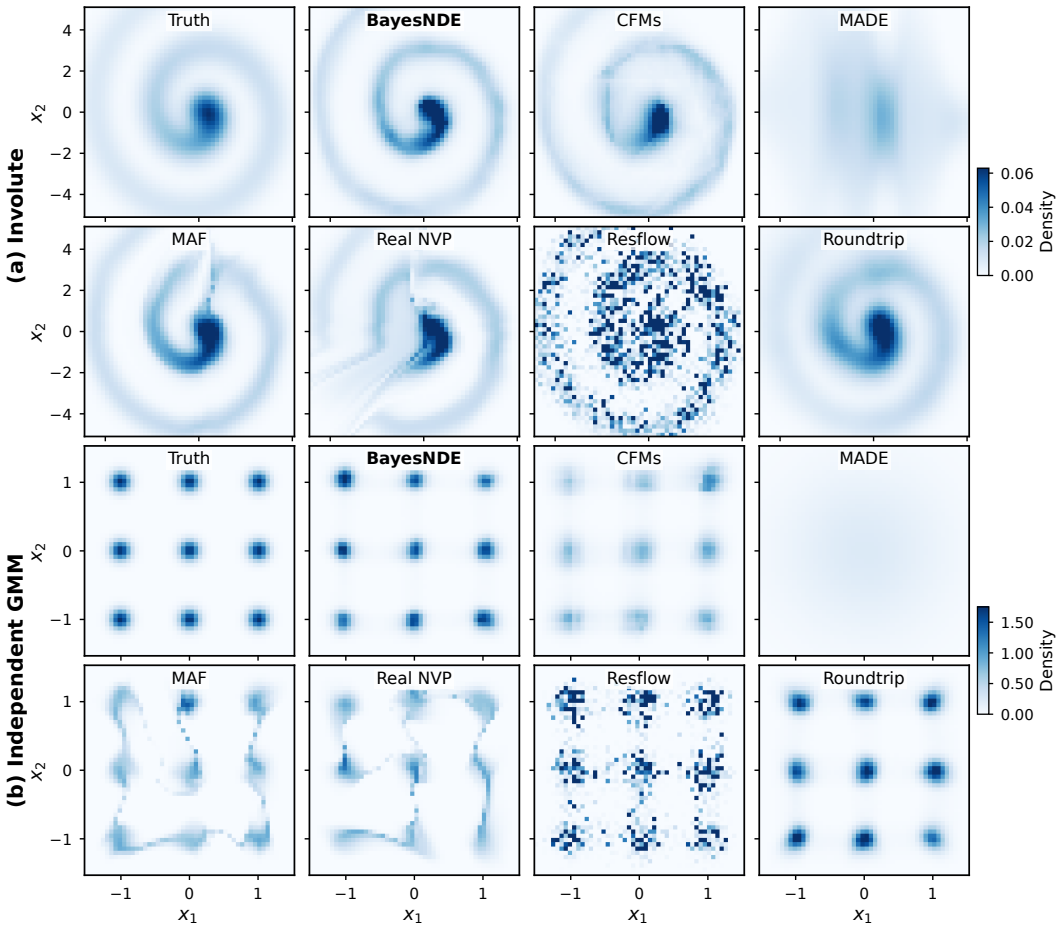


Figure 2: True and estimated densities for the two-dimensional involute distribution and independent Gaussian mixture.

5.3 REAL DATASETS

Table 1 reports average test log-likelihoods on the five real-world datasets, covering both unconditional and conditional density estimation. BayesNDE achieves the highest value on every dataset. The improvements are substantial across tasks. For instance, BayesNDE obtains the highest average test log-likelihood of 41.43 on BANK, compared with 35.74 for the best baseline Roundtrip.

On the three conditional datasets, BayesNDE improves over the strongest competing method by 3.68-19.01 nats. These consistent gains across datasets and task types demonstrate the effectiveness of BayesNDE for real-world density estimation.

5.4 Density-Based Downstream Tasks

We first evaluate the practical utility of BayesNDE for outlier detection on three ODDS benchmark datasets (Rayana, 2016): Shuttle, Cardio, and Pendigits. All density estimators were trained only on the training split, and test observations were ranked by estimated log-density, with lower-density observations more likely to be outliers. We compared BayesNDE with five generative density estimators as well as two dedicated anomaly-detection methods, one-class SVM (OC-SVM; Schölkopf et al., 2001) and Isolation Forest (I-Forest; Liu et al., 2008), using precision@k, which is the proportion of true anomalies among the k highest-ranked candidates.

Table 1: Precision@k on the ODDS anomaly-detection benchmarks, where k equals the number of outliers in the test split. Results are means over the same three random seeds for every method; OC-SVM is deterministic. Higher values are better.

Method	**Unconditional**		**Conditional**		
	BANK	**ParkTele**	**Pendigits10**	**Vehicle**	**EEGEye**
MADE	14.38±0.27	34.34±0.67	-54.56±0.96	-51.84±1.41	-50.25±0.18
MAF	-53.15±0.20	36.99±0.66	-52.29±0.78	-52.30±2.25	-49.47±0.21
Real NVP	27.21±0.38	36.45±0.68	-54.59±1.16	-63.03±3.61	-49.87±0.22
Resflow	23.68±0.18	36.25±0.63	-47.77±0.64	-45.97±1.13	-48.33±0.29
CFMs	10.23±0.07	16.32±0.19	-54.40±0.36	-49.65±0.87	-51.45±0.14
Roundtrip	35.74±0.13	49.64±0.33	-48.59±0.08	-40.10±0.70	-44.83±0.08
BayesNDE	**41.43±0.17**	**59.54±0.53**	**-28.76±0.44**	**-33.54±0.88**	**-41.15±0.19**

Table 2: Precision@k on the ODDS anomaly-detection benchmarks, averaged over three independent runs. Higher values are better.

	OC-SVM	I-Forest	Real NVP	MAF	CFMs	Resflow	Roundtrip	BayesNDE
Shuttle	0.953	0.958	0.735	0.861	0.932	0.477	0.958	**0.965**
Cardio	0.533	**0.556**	0.178	0.200	0.511	0.244	0.311	**0.556**
Pendigits	0.154	0.128	0.000	0.000	0.000	0.000	0.000	**0.180**

As shown in Table 2, BayesNDE achieves the highest precision@k on all three datasets, including 0.556 on Cardio and 0.180 on the challenging Pendigits dataset. These results show that the density estimates produced by BayesNDE are informative for downstream outlier detection and remain competitive with methods designed specifically for this task.

Next, we evaluate BayesNDE on the MNIST image dataset for conditional generation and classification. As shown in Figure 3, BayesNDE generates diverse and recognizable digits, with higher-density samples exhibiting more representative class-specific structure. With the uniform class prior $p(y) = 1/10$, we classify each test image using $\hat{y} = \arg\max_y p(y \mid \boldsymbol{x}) = \arg\max_y \{\log \widehat{p}_{\boldsymbol{\theta}}(\boldsymbol{x} \mid y) + \log p(y)\} = \arg\max_y \log \widehat{p}_{\boldsymbol{\theta}}(\boldsymbol{x} \mid y)$. BayesNDE achieves 98.6% test accuracy, compared with 98.0% for Roundtrip and 92.6% for MAF.

5.5 Ablation Analysis

We ablate the main components of BayesNDE on the synthetic mixture setting, using the same 500 held-out observations for all comparisons. Table 3 reports the density estimation accuracy. Bridge sampling attains higher ranking accuracy than importance sampling at every posterior budget, reaching the best accuracy importance sampling attains at any budget with four times fewer posterior samples (see details in Appendix A.4 S6). In the 2D ablation setting, removing the EGM warm start severely degrades density recovery. In 15 dimensions, removing correlation matching or the MMD terms also reduces density accuracy.

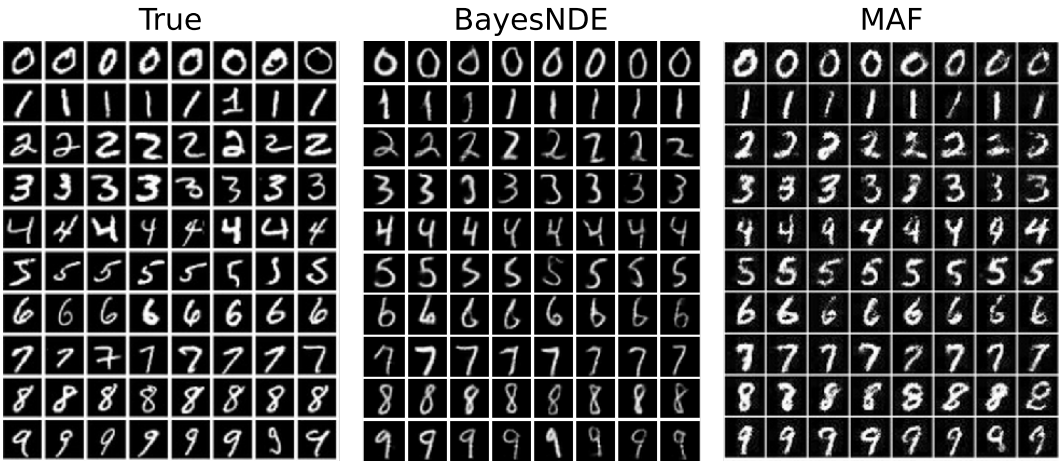
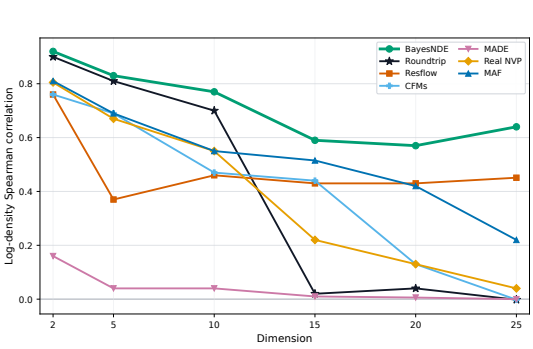
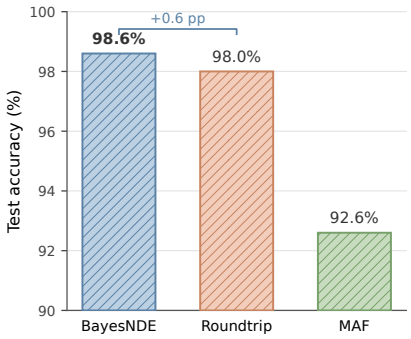


Figure 3: Conditional image generation on MNIST. The first column shows real MNIST examples; the remaining columns show samples generated by BayesNDE and MAF. For each digit class, representative generated images are shown in decreasing order of conditional density $p_{\boldsymbol{\theta}}(\boldsymbol{x} \mid y)$.



(a) Spearman rank correlation on independent GMM.



(b) Test accuracy on MNIST.

Figure 4: Empirical evaluation across benchmarks. (a) Density estimation accuracy across varying dimensions; (b) Classification performance on MNIST.

6 Discussion

We introduced BayesNDE, which casts pointwise density evaluation as posterior normalizing-constant estimation. Observation-specific posterior proposals and bridge sampling enable explicit density evaluation without invertible architectures or Jacobian determinants, while improving density estimation and downstream performance.

Its main limitation is running time: BayesNDE is slower than CFMs, MAF and Roundtrip (Appendix A.4). However, test-time posterior sampling and bridge estimation are independent across observations and can be further accelerated by parallel computing. Accuracy also depends on posterior exploration and proposal overlap in complex latent spaces. Future work will focus on faster training, posterior sampling and reusable proposals to reduce the cost of repeated density evaluation.

Table 3: Ablation results on synthetic data.

Method	Spearman ↑
BayesNDE (2D)	**0.9214**
w.o. EGM init.	0.0129
w.o. var. reg.	0.8929
BayesNDE (15D)	**0.6311**
w.o. BS	0.5932
w.o. EGM reg. (MMD)	0.5848
w.o. EGM reg. (Corr)	0.6169

AI USE STATEMENT

Generative AI tools were used to assist with language polishing and identifying grammatical and spelling errors. They were also used to check citation formatting, generate auxiliary LaTeX code for tables and figure layouts, and debug simple code errors. AI tools were consulted for possible hyperparameter choices, but these suggestions were not adopted in the reported experiments. AI was not used to develop the proposed methodology, prove mathematical claims, generate synthetic datasets, produce experimental results, or interpret the reported results. All AI-assisted content was reviewed and verified by the authors.

REPRODUCIBILITY STATEMENT

We provide an anonymized, installable source package as supplementary material and an anonymous repository link in the manuscript. We provide detailed descriptions of the BayesNDE model, posterior sampling, proposal construction, and bridge-sampling procedure in the main text. Experimental settings, model architectures, hyperparameters, data processing, and additional implementation details are provided in the appendix. All datasets used in our experiments are publicly available, with their sources and preprocessing procedures documented in the paper and appendix. Upon acceptance, the anonymous GitHub repository link will be replaced with a link to the public GitHub repository in the camera-ready version.

REFERENCES

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Ricky T. Q. Chen, Jens Behrmann, David Duvenaud, and Jörn-Henrik Jacobsen. Residual flows for invertible generative modeling. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using Real NVP. In *International Conference on Learning Representations*, 2017.

Dheeru Dua and Casey Graff. UCI machine learning repository. University of California, Irvine, School of Information and Computer Sciences, 2017. URL `https://archive.ics.uci.edu/ml`.

Simon Duane, A. D. Kennedy, Brian J. Pendleton, and Duncan Roweth. Hybrid monte carlo. *Physics Letters B*, 195(2):216–222, 1987. doi: 10.1016/0370-2693(87)91197-X.

Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. MADE: Masked autoencoder for distribution estimation. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 881–889. PMLR, 2015.

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

Quentin F. Gronau, Henrik Singmann, and Eric-Jan Wagenmakers. bridgesampling: An R package for estimating normalizing constants. *Journal of Statistical Software*, 92(10):1–29, 2020. doi: 10.18637/jss.v092.i10.

Tim Hesterberg. Weighted average importance sampling and defensive mixture distributions. *Technometrics*, 37(2):185–194, 1995. doi: 10.1080/00401706.1995.10484303.

Lennart Hoogerheide, Anne Opschoor, and Herman K. van Dijk. A class of adaptive importance sampling weighted EM algorithms for efficient and robust posterior and predictive simulation. *Journal of Econometrics*, 171(2):101–120, 2012. doi: 10.1016/j.jeconom.2012.06.011.

He Jia and Uros Seljak. Normalizing constant estimation with gaussianized bridge sampling. In *Symposium on Advances in Approximate Bayesian Inference*, pp. 1–14. PMLR, 2020.

Diederik P. Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*, volume 29, pp. 4743–4751, 2016.

Soheil Kolouri, Phillip E. Pope, Charles E. Martin, and Gustavo K. Rohde. Sliced-wasserstein autoencoders. In *International Conference on Learning Representations*, 2019.

Ilya Korsunsky, Nghia Millard, Jean Fan, Kamil Slowikowski, Fan Zhang, Kevin Wei, Yuriy Baglaenko, Michael Brenner, Po-Ru Loh, and Soumya Raychaudhuri. Fast, sensitive and accurate integration of single-cell data with harmony. *Nature Methods*, 16(12):1289–1296, 2019. doi: 10.1038/s41592-019-0619-0.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023.

Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pp. 413–422. IEEE, 2008. doi: 10.1109/ICDM.2008.17.

Qiao Liu and Wing Hung Wong. An AI-powered bayesian generative modeling approach for causal inference in observational studies. *Journal of the American Statistical Association*, 121(554): 976–987, 2026a. doi: 10.1080/01621459.2026.2654227.

Qiao Liu and Wing Hung Wong. An AI-powered bayesian generative modeling approach for arbitrary conditional inference, 2026b. URL https://arxiv.org/abs/2601.05355.

Qiao Liu, Jiaze Xu, Rui Jiang, and Wing Hung Wong. Density estimation using deep generative neural networks. *Proceedings of the National Academy of Sciences*, 118(15):e2101344118, 2021. doi: 10.1073/pnas.2101344118.

Qiao Liu, Zhongren Chen, and Wing Hung Wong. An encoding generative modeling approach to dimension reduction and covariate adjustment in causal inference with observational studies. *Proceedings of the National Academy of Sciences*, 121(23):e2322376121, 2024.

Guyue Luo and Qiao Liu. Bgm-iv: an ai-powered bayesian generative modeling approach for instrumental variable analysis. *arXiv preprint arXiv:2605.07029*, 2026.

Xiao-Li Meng and Wing Hung Wong. Simulating ratios of normalizing constants via a simple identity: A theoretical exploration. *Statistica Sinica*, 6(4):831–860, 1996.

Radford M. Neal. Mcmc using hamiltonian dynamics. In *Handbook of Markov Chain Monte Carlo*, pp. 113–162. Chapman and Hall/CRC, 2011. doi: 10.1201/b10905-6.

Art Owen and Yi Zhou. Safe and effective importance sampling. *Journal of the American Statistical Association*, 95(449):135–143, 2000. doi: 10.1080/01621459.2000.10473909.

George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*, volume 30, pp. 2338–2347, 2017.

Emanuel Parzen. On estimation of a probability density function and mode. *The Annals of Mathematical Statistics*, 33(3):1065–1076, 1962. doi: 10.1214/aoms/1177704472.

Shebuti Rayana. ODDS library. Stony Brook University, Department of Computer Science, 2016. URL https://shebuti.com/outlier-detection-datasets-odds/.

Bernhard Schölkopf, John C. Platt, John Shawe-Taylor, Alex J. Smola, and Robert C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13(7):1443–1471, 2001. doi: 10.1162/089976601750264965.

Alexander Tong, Kilian Fatras, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=CD9Snc73AW.

Aki Vehtari, Andrew Gelman, Daniel Simpson, Bob Carpenter, and Paul-Christian Bürkner. Rank-normalization, folding, and localization: An improved $\widehat{R}$ for assessing convergence of mcmc (with discussion). *Bayesian Analysis*, 16(2):667–718, 2021. doi: 10.1214/20-BA1221.

Jackie S. T. Wong, Jonathan J. Forster, and Peter W. F. Smith. Properties of the bridge sampler with a focus on splitting the mcmc sample. *Statistics and Computing*, 30(4):799–816, 2020. doi: 10.1007/s11222-019-09918-5.

Yuhuai Wu, Yuri Burda, Ruslan Salakhutdinov, and Roger Grosse. On the quantitative analysis of decoder-based generative models. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=B1M8JF9xx.

A Appendix

A.1 Proofs of theory

Proof of Theorem 1 We provide the technical conditions and proof for Theorem 1. Throughout, the observation x and the fitted generative parameter $\hat{\boldsymbol{\theta}}$ are fixed. Define

$$\pi_x(z) = p_{\hat{\boldsymbol{\theta}}}(x \mid z)p_Z(z), \qquad Z_x = \int \pi_x(z)\, dz, \qquad p_x(z) = \frac{\pi_x(z)}{Z_x}.$$

By construction, $Z_x = p_{\hat{\boldsymbol{\theta}}}(x)$.

Let $\mathcal{D}_{\text{fit}}$ denote the posterior sample used to construct the observation-specific proposal. Conditional on $\mathcal{D}_{\text{fit}}$, the fitted proposal q_x and the defensive proposal

$$\widetilde{q}_x(z) = (1-\epsilon)q_x(z) + \epsilon p_Z(z)$$

are treated as fixed. For notational simplicity, write $q(z) = \widetilde{q}_x(z)$.

Conditional on $\mathcal{D}_{\text{fit}}$, assume:

(A1) $q(z) > 0$ whenever $\pi_x(z) > 0$.

(A2) The proposal draws $\widetilde{z}_1, \ldots, \widetilde{z}_S$ are i.i.d. from q, and the held-out posterior draws $z_1^\star, \ldots, z_M^\star$ satisfy the law of large numbers under p_x for every bounded measurable function appearing below.

(A3) The bridge weights s_p and $s_q = 1 - s_p$ remain bounded away from zero: for some constant $\eta \in (0, 1/2)$,

$$\eta \le s_p, s_q \le 1 - \eta.$$

(A4) The fixed-point iteration is initialized by the importance-sampling estimator

$$\widehat{Z}_{\text{IS}} = \frac{1}{S}\sum_{s=1}^{S} \frac{\pi_x(\widetilde{z}_s)}{q(\widetilde{z}_s)},$$

and the iteration is run to a converged fixed point $\widehat{Z}_x$.

Condition (A2) allows the held-out posterior sample to be generated by an ergodic Markov chain; independence of the posterior draws is not required. When the proposal-fitting and bridge samples are obtained from a common Markov chain, (A2) is understood conditionally on $\mathcal{D}_{\text{fit}}$.

For $c > 0$, define

$$A_S(c) = \frac{1}{S}\sum_{s=1}^{S} \frac{\pi_x(\widetilde{z}_s)}{s_p\pi_x(\widetilde{z}_s) + s_q c\, q(\widetilde{z}_s)}, B_M(c) = \frac{1}{M}\sum_{j=1}^{M} \frac{q(z_j^\star)}{s_p\pi_x(z_j^\star) + s_q c\, q(z_j^\star)}.$$

The empirical bridge map is

$$\widehat{T}_{M,S}(c) = \frac{A_S(c)}{B_M(c)}.$$

Its population counterparts are

$$A(c) = \mathbb{E}_q\left[\frac{\pi_x(Z)}{s_p\pi_x(Z) + s_q c\, q(Z)}\right], B(c) = \mathbb{E}_{p_x}\left[\frac{q(Z)}{s_p\pi_x(Z) + s_q c\, q(Z)}\right].$$

We first establish the key population identity. Since $p_x(z) = \pi_x(z)/Z_x$,

$$\begin{aligned} B(c) &= \int \frac{q(z)}{s_p\pi_x(z) + s_q c\, q(z)} \frac{\pi_x(z)}{Z_x}\, dz \\ &= \frac{1}{Z_x}\int \frac{\pi_x(z)q(z)}{s_p\pi_x(z) + s_q c\, q(z)}\, dz \\ &= \frac{A(c)}{Z_x}. \end{aligned} \tag{13}$$

Consequently,

$$T(c) := \frac{A(c)}{B(c)} = Z_x \qquad \text{for every } c > 0. \tag{14}$$

Thus the population bridge map is constant: the choice of bridge iterate c and the fitted proposal affect Monte Carlo efficiency but not the population target.

We next show that the empirical bridge map converges uniformly to this population map on compact subsets of $(0, \infty)$. Let $K = [a, b] \subset (0, \infty)$ and define

$$f_c(z) = \frac{\pi_x(z)}{s_p\pi_x(z) + s_q c\, q(z)}, \qquad g_c(z) = \frac{q(z)}{s_p\pi_x(z) + s_q c\, q(z)}.$$

For all $c \in K$,

$$0 \le f_c(z) \le \frac{1}{s_p} \le \frac{1}{\eta}, 0 \le g_c(z) \le \frac{1}{s_q c} \le \frac{1}{\eta a}.$$

Hence both families are uniformly bounded.

Moreover, they are uniformly Lipschitz in c. Indeed,

$$\left|\frac{\partial f_c(z)}{\partial c}\right| = \frac{s_q\pi_x(z)q(z)}{\left[s_p\pi_x(z) + s_q c\, q(z)\right]^2}.$$

Using $4uv \le (u+v)^2$ with $u = s_p\pi_x(z)$ and $v = s_q c\, q(z)$ gives

$$\left|\frac{\partial f_c(z)}{\partial c}\right| \le \frac{1}{4s_p a} \le \frac{1}{4\eta a}.$$

Similarly,

$$\left|\frac{\partial g_c(z)}{\partial c}\right| = \frac{s_q q(z)^2}{\left[s_p\pi_x(z) + s_q c\, q(z)\right]^2} \le \frac{1}{s_q a^2} \le \frac{1}{\eta a^2}.$$

Therefore the classes $\{f_c : c \in K\}$ and $\{g_c : c \in K\}$ are uniformly bounded and equicontinuous in their one-dimensional parameter c.

By the law of large numbers in (A2), convergence holds for every fixed c. A finite-grid argument together with the preceding uniform Lipschitz bounds then yields

$$\sup_{c\in K} |A_S(c) - A(c)| \xrightarrow{p} 0 \tag{15}$$

and

$$\sup_{c\in K} |B_M(c) - B(c)| \xrightarrow{p} 0. \tag{16}$$

It remains to control the denominator. By (A1), the integrand defining $A(c)$ is strictly positive on the support of π_x. Hence $A(c) > 0$ for every $c > 0$. Since $A(c)$ is continuous in c, compactness of K gives $\inf_{c\in K} A(c) > 0$. From equation 13,

$$\inf_{c\in K} B(c) = \frac{1}{Z_x}\inf_{c\in K} A(c) > 0.$$

Combining this fact with equation 15–equation 16 and the continuous mapping theorem gives

$$\sup_{c\in K}\left|\widehat{T}_{M,S}(c) - Z_x\right| \xrightarrow{p} 0. \tag{17}$$

We now connect the uniform convergence result to the actual BayesNDE iteration. First, the importance-sampling initializer is consistent. Indeed, by (A1),

$$\mathbb{E}_q\left[\frac{\pi_x(Z)}{q(Z)}\right] = \int \pi_x(z)\,dz = Z_x,$$

and the random variable $\pi_x(Z)/q(Z)$ is integrable because its expectation equals the finite constant Z_x. Therefore, by the law of large numbers,

$$\widehat{Z}_{\mathrm{IS}} \xrightarrow{p} Z_x. \tag{18}$$

Choose, for example,

$$K = \left[\frac{Z_x}{2}, \frac{3Z_x}{2}\right].$$

By equation 18, $\widehat{Z}_{\mathrm{IS}} \in K$ with probability tending to one. Furthermore, by equation 17,

$$\sup_{c\in K}\left|\widehat{T}_{M,S}(c) - Z_x\right| < \frac{Z_x}{4}$$

with probability tending to one. On this event,

$$\widehat{T}_{M,S}(K) \subset \left[\frac{3Z_x}{4}, \frac{5Z_x}{4}\right] \subset K.$$

Thus, once initialized in K, every subsequent bridge iterate remains in K. If the iteration converges to the reported fixed point $\widehat{Z}_x$, then

$$\widehat{Z}_x = \widehat{T}_{M,S}(\widehat{Z}_x), \qquad \widehat{Z}_x \in K,$$

and hence

$$|\widehat{Z}_x - Z_x| \le \sup_{c\in K}\left|\widehat{T}_{M,S}(c) - Z_x\right| \xrightarrow{p} 0.$$

Therefore,

$$\widehat{Z}_x \xrightarrow{p} Z_x = p_{\hat{\boldsymbol{\theta}}}(x).$$

□

Proof of Theorem 2

Fix x and abbreviate

$$p(z) = p_{\hat{\boldsymbol{\theta}}}(z \mid x), \qquad q(z) = \widetilde{q}_x(z), \qquad Z = Z_x,$$

so that $\pi_x(z) = Zp(z)$. Here

$$\widetilde{q}_x(z) = (1-\epsilon)q_x(z) + \epsilon p_Z(z)$$

is the defensive proposal used by BayesNDE. Let $N = M_b + S$, with $M_b/N \to s_p \in (0,1)$ and $S/N \to s_q = 1 - s_p$.

For independent posterior and proposal samples, the asymptotically optimal bridge function of Meng & Wong (1996) is proportional to

$$h^\star(\boldsymbol{z}) = \frac{1}{s_p\pi_x(z) + s_q Z q(z)} \propto \frac{1}{s_p p(z) + s_q q(z)}.$$

Define

$$d(z) = s_p p(z) + s_q q(z), \qquad \mathcal{O} = \int \frac{p(z)q(z)}{d(z)}\,dz.$$

Under $h^\star(\boldsymbol{z})$, the relative bridge estimator can be written as

$$\frac{\widehat{Z}}{Z} = \frac{\frac{1}{S}\sum_{s=1}^{S} U(\widetilde{z}_s)}{\frac{1}{M_b}\sum_{j=1}^{M_b} V(z_j^\star)}, \qquad U(z) = \frac{p(z)}{d(z)}, \quad V(z) = \frac{q(z)}{d(z)}. \tag{19}$$

Since $\mathbb{E}_q[U] = \mathbb{E}_p[V] = \mathcal{O}$, the joint central limit theorem gives

$$\sqrt{N}\begin{pmatrix} \overline{U}_S - \mathcal{O} \\ \overline{V}_{M_b} - \mathcal{O} \end{pmatrix} \xrightarrow{d} \mathcal{N}\left(0, \begin{pmatrix} \mathrm{Var}_q(U)/s_q & 0 \\ 0 & \mathrm{Var}_p(V)/s_p \end{pmatrix}\right).$$

Applying the delta method to $g(a,b) = a/b$ at $(\mathcal{O}, \mathcal{O})$ yields

$$\sqrt{N}\left(\frac{\widehat{Z}}{Z} - 1\right) \xrightarrow{d} \mathcal{N}(0, \sigma^2), \qquad \sigma^2 = \frac{1}{\mathcal{O}^2}\left[\frac{\mathrm{Var}_q(U)}{s_q} + \frac{\mathrm{Var}_p(V)}{s_p}\right]. \tag{20}$$

The variance simplifies using

$$\begin{aligned} s_p\mathbb{E}_q[U^2] + s_q\mathbb{E}_p[V^2] &= \int \frac{s_p p(z)^2 q(z) + s_q p(z) q(z)^2}{d(z)^2}\, dz \\ &= \int \frac{p(z)q(z)}{d(z)}\, dz = \mathcal{O}. \end{aligned} \tag{21}$$

Together with $\mathbb{E}_q[U] = \mathbb{E}_p[V] = \mathcal{O}$ and $s_p + s_q = 1$, this gives

$$\frac{\mathrm{Var}_q(U)}{s_q} + \frac{\mathrm{Var}_p(V)}{s_p} = \frac{\mathcal{O} - \mathcal{O}^2}{s_p s_q}.$$

Substitution into equation 20 therefore yields

$$\sqrt{M_b + S}\left(\frac{\widehat{Z}_x}{Z_x} - 1\right) \xrightarrow{d} \mathcal{N}\left(0, \frac{1}{s_p s_q}\left[\mathcal{O}_x(\widetilde{q}_x)^{-1} - 1\right]\right).$$

It remains to characterize the overlap. First, for fixed $p(z) \geq 0$, define

$$f_z(q) = \frac{p(z)q}{s_p p(z) + s_q q}, \qquad q \geq 0.$$

For $p(z) > 0$,

$$\frac{\partial f_z(q)}{\partial q} = \frac{s_p p(z)^2}{\left[s_p p(z) + s_q q\right]^2} \geq 0,$$

so the overlap integrand is nondecreasing in its second argument. Because the defensive proposal satisfies

$$\widetilde{q}_x(z) = (1-\epsilon)q_x(z) + \epsilon p_Z(z) \geq \epsilon p_Z(z)$$

pointwise, it follows that

$$\mathcal{O}_x(\widetilde{q}_x) \geq \mathcal{O}_x(\epsilon p_Z).$$

Moreover, under the Gaussian latent prior used by BayesNDE, $p_Z(z) > 0$ for all z, and hence $\mathcal{O}_x(\epsilon p_Z) > 0$.

For the upper bound, by the weighted harmonic–arithmetic mean inequality,

$$\frac{p(z)q(z)}{s_p p(z) + s_q q(z)} = \left(\frac{s_p}{q(z)} + \frac{s_q}{p(z)}\right)^{-1} \leq s_p q(z) + s_q p(z),$$

with the left-hand side defined as zero whenever $p(z)q(z) = 0$. Integrating gives

$$\mathcal{O}_x(q) \leq s_p + s_q = 1.$$

Equality holds if and only if $p(z) = q(z)$ almost everywhere. Taking $q = \widetilde{q}_x$ therefore gives

$$0 < \mathcal{O}_x(\epsilon p_Z) \leq \mathcal{O}_x(\widetilde{q}_x) \leq 1,$$

with the upper bound attained if and only if $\widetilde{q}_x = p_x$ almost everywhere.

A.2 DATASETS IN EXPERIMENTS

S1. Data Generating Processes for Synthetic Distributions

Independent Gaussian mixture. Following (Liu et al., 2021), we consider an independent Gaussian mixture (Independent GMM) distribution and extend the original two-dimensional construction to p dimensions. Specifically, we generate $\mathbf{X} = (X_1, \ldots, X_p)^\top$ with mutually independent coordinates, where each coordinate follows the same three-component Gaussian mixture, $X_j \sim \frac{1}{3}\mathcal{N}(-1, 0.1^2) + \frac{1}{3}\mathcal{N}(0, 0.1^2) + \frac{1}{3}\mathcal{N}(1, 0.1^2)$, for $j = 1, \ldots, p$. Equivalently, the joint density is $p(\mathbf{x}) = \prod_{j=1}^{p} \left[\frac{1}{3}\sum_{\mu\in\{-1,0,1\}} \phi(x_j; \mu, 0.1^2)\right]$, where $\phi(\cdot; \mu, \sigma^2)$ denotes the Gaussian density with mean μ and variance σ^2. This construction yields 3^p equally weighted modes located at the Cartesian product $\{-1, 0, 1\}^p$. As p increases, the number of modes therefore grows exponentially, providing an increasingly challenging benchmark for evaluating the ability of density estimators to recover highly multimodal distributions in higher dimensions.

Involute distribution. The involute distribution is constructed by first sampling a latent variable $R \sim \mathrm{Uniform}(0, 2\pi)$ and then independently generating $X_1 \mid R = r \sim \mathcal{N}(r\sin(2r), 0.4^2)$ and $X_2 \mid R = r \sim \mathcal{N}(r\cos(2r), 0.4^2)$. Thus, conditional on $R = r$, the observation is centered along the nonlinear curve $(r\sin(2r), r\cos(2r))$ with isotropic Gaussian perturbation. The corresponding marginal density is $p(\mathbf{x}) = \frac{1}{2\pi}\int_0^{2\pi} \phi(x_1; r\sin(2r), 0.4^2)\phi(x_2; r\cos(2r), 0.4^2)\, dr$, which can be evaluated numerically when the ground-truth density is required. In contrast to the isolated modes of Independent GMM, this distribution concentrates probability mass around a highly nonlinear curved structure and is therefore used to assess the ability of density estimators to capture complex nonlinear geometries.

S2. Details of data preprocessing for the outlier-detection datasets used in our study.

Shuttle. The Shuttle dataset (`http://odds.cs.stonybrook.edu/shuttle-dataset/`) is derived from the Statlog Shuttle classification dataset and contains nine numerical features. The original training and test sets were combined before constructing the outlier-detection benchmark. Samples from the five smallest classes, namely classes 2, 3, 5, 6, and 7, were grouped together as outliers, while samples from class 1 were treated as inliers. Samples from class 4 were discarded. The resulting dataset contains 49,097 observations, including 3,511 outliers (7.15%).

Cardio. The Cardio dataset (`http://odds.cs.stonybrook.edu/cardio-dataset/`) is derived from the UCI Cardiotocography dataset and consists of 21 numerical features extracted from fetal-heart-rate and uterine-contraction measurements. The original observations were classified by expert obstetricians into normal, suspect, and pathologic fetal states. For the ODDS outlier-detection benchmark, the suspect class was discarded, the 1,655 normal observations were treated as inliers, and the 176 pathologic observations were treated as outliers. The resulting dataset contains 1,831 observations with an outlier proportion of 9.61%.

Pendigits. The Pendigits dataset (`http://odds.cs.stonybrook.edu/pendigits-dataset/`) is derived from the UCI Pen-Based Recognition of Handwritten Digits dataset. Each observation is represented by 16 numerical features corresponding to eight sampled two-dimensional coordinates along a handwritten pen trajectory. The original dataset contains ten digit classes, from 0 to 9. In the ODDS benchmark, digit 0 was designated as the outlier class and downsampled to 156 observations, whereas the 6,714 observations corresponding to digits 1–9 were treated as inliers. The resulting dataset contains 6,870 observations with an outlier proportion of 2.27%.

Common preprocessing and data splitting. For each dataset, every feature was transformed to the interval $[0, 1]$ using feature-wise minimum and maximum values. The observations and their labels were then jointly permuted using NumPy random seed 0. After permutation, the final 10% of the observations were retained as the test set. Of the remaining observations, the final 10% were used for validation and the rest were used for training. This produces an approximately 81/9/10 training/validation/test split. The resulting split sizes are reported in Table **??**. Class labels were not provided as inputs to any of the density-estimation models.

Table 4: Basic descriptions of the three datasets used for outlier detection in our study.

Dataset	Dim(z)	Dim(x)	Outliers (%)	Test outliers (k)	# examples		
					Train	Validation	Test
Shuttle	3	9	7.15	365	39770	4418	4909
Cardio	5	21	9.61	15	1484	164	183
Pendigits	5	16	2.27	13	5565	618	687

Outliers (%) is the outlier rate of the full dataset; Test outliers (k) is the number of outliers in the test split, which sets k in precision@k. Dim(z) denotes the latent dimension used by Roundtrip. For BayesNDE, we set Dim(z) = Dim(x), corresponding to 9, 21, and 16 for Shuttle, Cardio, and Pendigits, respectively.

S3. Details of data preprocessing for the Real word datasets used in our study.

BANK. The BANK dataset relates to a marketing campaign of a Portuguese banking institution, where the goal is to predict whether the client will subscribe a deposit. Label encoding was used for discrete features in the raw data with values between 0 and `n_classes`. Then a uniform noise of $(-0.2, 0.2)$ was added to each feature. At last, the raw data was applied a feature scaling through a min–max normalization and randomly split into 90% training set and 10% test. Note that for neural density estimators, 10% of the training set is kept for validation. BANK is used for unconditional density estimation.

ParkTele. The Parkinsons Telemonitoring dataset (UCI archive 189) contains biomedical voice measurements from 42 patients with early-stage Parkinson's disease, recruited for a six-month trial of a telemonitoring device. Each example comprises 16 dysphonia measures derived from a sustained phonation: five jitter variants, six shimmer variants, the noise-to-harmonics and harmonics-to-noise ratios, and the nonlinear measures RPDE, DFA and PPE. Two of the 16 were removed because they are definitional multiples of other columns: `Jitter:DDP` equals $3\times$`Jitter:RAP` and `Shimmer:DDA` equals $3\times$`Shimmer:APQ3` (the observed ratios lie in $[2.9697, 3.0294]$ and $[2.9942, 3.0062]$, departing from exactly three only through the five-decimal rounding of the published file). Retaining them makes the design matrix numerically singular, so that a continuous density on $\mathbb{R}^{16}$ is not well defined. The remaining 14 features were scaled by a per-feature min–max normalization, and the same split as the BANK dataset was used. The subject identifier, age, sex, test time and the two UPDRS scores were discarded. ParkTele is used for unconditional density estimation. We note that the 5,875 recordings come from 42 subjects, so the random split places recordings of the same subject in both the training and the test set; all methods share the identical split.

Pendigits10. The Pen-Based Recognition of Handwritten Digits dataset (UCI archive 81) contains pen trajectories collected from 44 writers on a pressure-sensitive tablet. Each digit is resampled to eight equally spaced points along the trajectory, giving 16 integer coordinates in $[0, 100]$, and is labelled with one of the ten digit classes. Because a continuous density is not defined on an integer lattice, a uniform noise of $(0, 1)$ was added to each coordinate with a fixed random seed, so that all methods see identical inputs; with unit lattice spacing, the expected log-density under uniform dequantization provides a lower bound on the discrete log-probability mass by Jensen's inequality.

Vehicle. The Statlog Vehicle Silhouettes dataset (UCI archive 149) describes four vehicle types (bus, Opel, Saab, van) by 18 integer-valued shape descriptors extracted from silhouette images, including compactness, circularity, radius ratio, the axis-aligned and principal-axis moments, scatter ratio, elongatedness, hollows ratio and the skewness and kurtosis about the major and minor axes. A uniform noise of $(0, 1)$ was added to each feature for the same reason as in Pendigits10.

EEGEye. The EEG Eye State dataset (UCI archive 264) records one continuous 117-second measurement from an Emotiv EEG headset, giving 14 electrode channels per time point, with the eye state (open or closed) determined from a simultaneous video recording. Four of the 14,980 examples are documented measurement artifacts in which individual channels reach values above 7×10^5 against a median near 4.3×10^3; these were removed using the criterion that the robust z-score $|x - \text{median}|/\text{MAD}$ exceeds 100 in any channel, leaving 14,976 examples.

MNIST. MNIST (http://yann.lecun.com/exdb/mnist/) contains 28×28 grayscale images of handwritten digits. The images were flattened into 784-dimensional vectors and scaled to $[0, 1]$.

The descriptions of the five tabular datasets and the image dataset (MNIST), including feature dimension and sample size, are summarized in Table 5.

Table 5: Basic descriptions of the 6 datasets for density estimation used in our study. BANK and ParkTele are used for unconditional density estimation; Pendigits10, Vehicle and EEGEye for conditional density estimation, where $\dim(\boldsymbol{x})$ counts only the continuous features and the class label is the conditioning variable.

Dataset	Domain	Dim($\boldsymbol{z}$)	Dim($\boldsymbol{x}$)	# of examples		
				Train	Validation	Test
Vehicle	Transportation	6	18	684	77	85
ParkTele	Medicine	8	14	4,760	528	587
Pendigits10	Handwriting	8	16	8,902	990	1,100
EEGEye	Neuroscience	5	14	12,130	1,348	1,498
BANK	Finance	8	17	36,621	4,069	4,521
MNIST	Image	10	784	50,000	10,000	10,000

A.3 Conditional Density Estimation

BayesNDE extends directly to conditional density estimation when an observed conditioning variable y is available. We use a conditional generative model

$$p_{\boldsymbol{\theta}}(x \mid z, y), \qquad z \sim p_Z(z),$$

where y is provided as an additional input to the generative network. For a fitted model $\hat{\boldsymbol{\theta}}$, the conditional density of interest is

$$p_{\hat{\boldsymbol{\theta}}}(x \mid y) = \int p_{\hat{\boldsymbol{\theta}}}(x \mid z, y) p_Z(z)\, dz.$$

Thus, defining

$$\pi_{x,y}(z) = p_{\hat{\boldsymbol{\theta}}}(x \mid z, y) p_Z(z),$$

its normalizing constant is

$$Z_{x|y} = \int \pi_{x,y}(z)\, dz = p_{\hat{\boldsymbol{\theta}}}(x \mid y),$$

and the corresponding latent posterior is

$$p_{\hat{\boldsymbol{\theta}}}(z \mid x, y) = \frac{\pi_{x,y}(z)}{Z_{x|y}}.$$

Conditional density evaluation therefore follows exactly the same procedure as in the unconditional case. HMC targets the unnormalized log posterior $\log p_{\hat{\boldsymbol{\theta}}}(x \mid z, y) + \log p_Z(z)$; posterior samples are split into proposal-fitting and bridge subsets; an observation-specific proposal $q_{x,y}(z)$ is fitted from the former; and bridge sampling estimates $Z_{x|y}$. No modification of the bridge-sampling estimator is required beyond replacing π_x by $\pi_{x,y}$ throughout. For categorical conditioning variables used in our experiments, y is one-hot encoded and supplied to the conditional generative network.

A.4 Implementation

S1. Details of network regimes for the independent gussian datasets used in our simulation study.

We used two network regimes determined by the observed dimension p. For low-dimensional data ($p \leq 10$), the generator G and encoder E were fully connected multilayer perceptrons with five hidden layers of width 256. For higher-dimensional data ($p > 10$), G and E were residual networks

Table 6: Network architecture used by BayesNDE for conditional MNIST image generation and density estimation. Images are scaled to $[0,1]^{784}$, the class label $y \in \{0,1\}^{10}$ is one-hot encoded, and the latent variable is $z \in \mathbb{R}^{10}$. For convolutional layers, k, s, and c denote kernel size, stride, and number of output channels, respectively. All convolutions use same padding. BN denotes batch normalization and LReLU has negative slope 0.2. The decoder has two output heads that parameterize a Gaussian distribution in logit space.

Conditional decoder $G_{\boldsymbol{\theta}}$	**Image discriminator D_x**
Inputs: $z \in \mathbb{R}^{10}$ and $y \in \{0,1\}^{10}$	Inputs: $x \in [0,1]^{784}$ and $y \in \{0,1\}^{10}$
$\mathrm{Concat}(z, y) \in \mathbb{R}^{20}$	Reshape x to $28 \times 28 \times 1$; tile y to $28 \times 28 \times 10$
FC 6272, LReLU	$\mathrm{Concat}(x, \mathrm{tile}(y))$: $28 \times 28 \times 11$
Reshape to $7 \times 7 \times 128$	Conv ($k=5$, $s=2$, $c=32$), LReLU: $14 \times 14 \times 32$
Tile y to $7 \times 7 \times 10$ and concatenate: $7 \times 7 \times 138$	Conv ($k=5$, $s=2$, $c=64$), LReLU: $7 \times 7 \times 64$
Transposed conv ($k=3$, $s=2$, $c=64$), BN, LReLU: $14 \times 14 \times 64$	Conv ($k=3$, $s=2$, $c=128$), LReLU: $4 \times 4 \times 128$
Transposed conv ($k=3$, $s=2$, $c=32$), BN, LReLU: $28 \times 28 \times 32$	Flatten to 2048; concatenate y: 2058
Conv ($k=3$, $s=1$, $c=32$), BN, LReLU	FC 128, LReLU
Two parallel Conv heads ($k=1$, $s=1$, $c=1$)	FC 1 (linear score)
Flatten heads to $m_{\boldsymbol{\theta}}(z,y), r_{\boldsymbol{\theta}}(z,y) \in \mathbb{R}^{784}$	
$v_{\boldsymbol{\theta}} = v_{\min} + (v_{\max} - v_{\min})[1 - \exp\{-\mathrm{softplus}(r_{\boldsymbol{\theta}})\}]$	
$v_{\min} = 10^{-4}$ and $v_{\max} = 4$	
Conditional encoder E_ϕ	**Latent discriminator D_z**
Inputs: $x \in [0,1]^{784}$ and $y \in \{0,1\}^{10}$	Input: $z \in \mathbb{R}^{10}$
Reshape x to $28 \times 28 \times 1$; tile y to $28 \times 28 \times 10$	FC 256, BN, Tanh
$\mathrm{Concat}(x, \mathrm{tile}(y))$: $28 \times 28 \times 11$	FC 256, BN, Tanh
Conv ($k=3$, $s=2$, $c=32$), BN, LReLU: $14 \times 14 \times 32$	FC 128, BN, Tanh
Conv ($k=3$, $s=2$, $c=64$), BN, LReLU: $7 \times 7 \times 64$	FC 64, BN, Tanh
Conv ($k=3$, $s=1$, $c=128$), BN, LReLU: $7 \times 7 \times 128$	FC 1 (linear score)
Flatten to 6272; concatenate y: 6282	
FC 256, LReLU	
FC 10 (linear latent code)	
For image generation, a logit $L \sim \mathcal{N}(m_{\boldsymbol{\theta}}, \mathrm{diag}(v_{\boldsymbol{\theta}}))$ is drawn and the displayed image is $\mathrm{sigmoid}(L)$.	

with five width-256 hidden blocks. The data-space and latent-space discriminators used hidden widths 256–256–128–64 in both regimes. Hidden layers used LeakyReLU activations.

S2. EGM initialization

We initialize the generator G, encoder E, and discriminators D_x and D_z using encoding generative modeling (EGM). The EGM objective is

$$\begin{aligned}\mathcal{L}_{\mathrm{EGM}} = {} & \lambda_{\mathrm{adv},x}\mathcal{L}_{\mathrm{adv},x} + \lambda_{\mathrm{adv},z}\mathcal{L}_{\mathrm{adv},z} + \lambda_{\mathrm{rec},x}\mathcal{L}_{\mathrm{rec},x} + \lambda_{\mathrm{rec},z}\mathcal{L}_{\mathrm{rec},z} \\ & + \lambda_{\mathrm{corr}}\mathcal{L}_{\mathrm{corr}} + \lambda_{\mathrm{var}}\mathcal{L}_{\mathrm{logvar}} + \lambda_{\mathrm{MMD,marg}}\mathcal{L}_{\mathrm{MMD,marg}} \\ & + \lambda_{\mathrm{MMD,joint}}\mathcal{L}_{\mathrm{MMD,joint}} + \lambda_{\mathrm{SW}}\mathcal{L}_{\mathrm{SW}},\end{aligned} \tag{22}$$

Adversarial losses. The discriminators minimize the least-squares objective

$$\begin{aligned}\mathcal{L}_D = {} & \tfrac{1}{2}\Big\{\mathbb{E}_{\boldsymbol{x}\sim P^*}\big[(0.9 - D_x(\boldsymbol{x}))^2\big] + \mathbb{E}_{\boldsymbol{z}\sim P_Z}\big[(0.1 - D_x(G(\boldsymbol{z})))^2\big]\Big\} \\ & + \tfrac{1}{2}\Big\{\mathbb{E}_{\boldsymbol{z}\sim P_Z}\big[(0.9 - D_z(\boldsymbol{z}))^2\big] + \mathbb{E}_{\boldsymbol{x}\sim P^*}\big[(0.1 - D_z(E(\boldsymbol{x})))^2\big]\Big\},\end{aligned} \tag{23}$$

while G and E minimize

$$\mathcal{L}_{\mathrm{adv},x} = \mathbb{E}_{\boldsymbol{z}\sim P_Z}\big[(0.9 - D_x(G(\boldsymbol{z})))^2\big], \qquad \mathcal{L}_{\mathrm{adv},z} = \mathbb{E}_{\boldsymbol{x}\sim P^*}\big[(0.9 - D_z(E(\boldsymbol{x})))^2\big]. \tag{24}$$

Discriminator and generator/encoder updates alternate one-to-one.

Reconstruction losses.

$$\mathcal{L}_{\mathrm{rec},x} = \mathbb{E}_{\boldsymbol{x}\sim P^*}\left[\tfrac{1}{p}\big\|\boldsymbol{x} - G(E(\boldsymbol{x}))\big\|_2^2\right], \qquad \mathcal{L}_{\mathrm{rec},z} = \mathbb{E}_{\boldsymbol{z}\sim P_Z}\left[\tfrac{1}{p}\big\|\boldsymbol{z} - E(G(\boldsymbol{z}))\big\|_2^2\right]. \tag{25}$$

Correlation matching. The correlation matching is

$$\mathcal{L}_{\mathrm{corr}} = \frac{1}{p^2}\sum_{i=1}^{p}\sum_{j=1}^{p}\left[\mathrm{Corr}(\boldsymbol{x}'_i, \boldsymbol{x}'_j) - \mathrm{Corr}(\boldsymbol{x}_i, \boldsymbol{x}_j)\right]^2, \qquad \boldsymbol{x}' = G(E(\boldsymbol{x})). \tag{26}$$

Marginal MMD. With the one-dimensional Gaussian kernel $k_h(u,v) = \exp\{-(u-v)^2/(2h^2)\}$ and bandwidths $\mathcal{H} = \{0.05, 0.1, 0.2, 0.5, 1.0\}$,

$$\mathcal{L}_{\mathrm{MMD,marg}} = \frac{1}{|\mathcal{H}|}\sum_{h\in\mathcal{H}}\frac{1}{p}\sum_{j=1}^{p}\frac{1}{n^2}\sum_{a,b=1}^{n}\Big[k_h(x_{aj}, x_{bj}) + k_h(\tilde{x}_{aj}, \tilde{x}_{bj}) - 2k_h(x_{aj}, \tilde{x}_{bj})\Big]. \tag{27}$$

Joint MMD. Both samples are first standardized coordinatewise by the pooled minibatch mean $\boldsymbol{m}$ and variance $\boldsymbol{v}$ (treated as constants), $\bar{\boldsymbol{x}} = (\boldsymbol{x} - \boldsymbol{m})/\sqrt{\boldsymbol{v} + 10^{-6}}$, and likewise $\bar{\tilde{\boldsymbol{x}}}$. With $\kappa_s(\boldsymbol{u}, \boldsymbol{v}) = \exp\{-\|\boldsymbol{u} - \boldsymbol{v}\|_2^2/(2sp)\}$ and $s \in \mathcal{S} = \{0.5, 1, 2\}$,

$$\mathcal{L}_{\mathrm{MMD,joint}} = \max\Big\{0,\ \frac{1}{|\mathcal{S}|}\sum_{s\in\mathcal{S}}\frac{1}{n^2}\sum_{a,b=1}^{n}\left[\kappa_s(\bar{\boldsymbol{x}}_a, \bar{\boldsymbol{x}}_b) + \kappa_s(\bar{\tilde{\boldsymbol{x}}}_a, \bar{\tilde{\boldsymbol{x}}}_b) - 2\kappa_s(\bar{\boldsymbol{x}}_a, \bar{\tilde{\boldsymbol{x}}}_b)\right]\Big\}. \tag{28}$$

Sliced Wasserstein. Using the same standardization, let $\Theta = [\boldsymbol{e}_1, \ldots, \boldsymbol{e}_p, \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_{L-p}]$ with $L = 64$, where the $\boldsymbol{e}_j$ are coordinate axes and the $\boldsymbol{\theta}_l$ are fixed random unit vectors. For each direction $\boldsymbol{\omega}_l$, let $u_{(1)}^{(l)} \le \cdots \le u_{(n)}^{(l)}$ and $v_{(1)}^{(l)} \le \cdots \le v_{(n)}^{(l)}$ be the sorted projections $\boldsymbol{\omega}_l^\top \bar{\boldsymbol{x}}_a$ and $\boldsymbol{\omega}_l^\top \bar{\tilde{\boldsymbol{x}}}_a$. Then

$$\mathcal{L}_{\mathrm{SW}} = \frac{1}{nL}\sum_{l=1}^{L}\sum_{i=1}^{n}\big(u_{(i)}^{(l)} - v_{(i)}^{(l)}\big)^2. \tag{29}$$

The conditional-variance stabilization term is

$$\mathcal{L}_{\mathrm{logvar}} = \mathbb{E}_{\boldsymbol{z}\sim P_Z}\left[\frac{1}{p}\sum_{j=1}^{p}\big\{\log\big(\sigma_j^2(\boldsymbol{z}) + 10^{-8}\big) - \log(0.01)\big\}^2\right]. \tag{30}$$

Throughout, the generator is stochastic, $G(\boldsymbol{z}) = \boldsymbol{\mu}(\boldsymbol{z}) + \boldsymbol{\sigma}(\boldsymbol{z}) \odot \boldsymbol{\epsilon}$, $\boldsymbol{\epsilon} \sim N(\boldsymbol{0}, I_p)$, with $\sigma_j^2(\boldsymbol{z}) = \mathrm{softplus}(\cdot) + 10^{-6}$, and $P_Z = N(\boldsymbol{0}, I_p)$. All expectations are estimated on minibatches of size $n = 256$; we write $\{\boldsymbol{x}_a\}_{a=1}^n \sim P^*$ and $\{\tilde{\boldsymbol{x}}_a = G(\boldsymbol{z}_a)\}_{a=1}^n$, $\boldsymbol{z}_a \sim P_Z$. For the high-dimensional independent-GMM experiments, $\lambda_{\mathrm{adv},x} = \lambda_{\mathrm{adv},z} = 1$, $\lambda_{\mathrm{rec},x} = 3$, $\lambda_{\mathrm{rec},z} = 1$, $\lambda_{\mathrm{corr}} = 0.3$, $\lambda_{\mathrm{MMD,marg}} = 1000$, $\lambda_{\mathrm{MMD,joint}} = 100$, $\lambda_{\mathrm{SW}} = 300$, and $\lambda_{\mathrm{var}} = 0$. For the low-dimensional independent-GMM experiments, G and E are five-layer width-256 multilayer perceptrons and the EGM learning rate is 10^{-3}. The final configurations use $\lambda_{\mathrm{adv},x} = \lambda_{\mathrm{adv},z} = \lambda_{\mathrm{rec},z} = 1$, $\lambda_{\mathrm{corr}} = 0.3$, and $\lambda_{\mathrm{var}} = 0.01$. The data-space reconstruction weight is $\lambda_{\mathrm{rec},x} = 1$ for the 2D settings and $\lambda_{\mathrm{rec},x} = 3$ for the 5D and 10D settings. The marginal-MMD penalty is disabled except in the 5D configuration, where $\lambda_{\mathrm{MMD,marg}} = 100$.

For the five real datasets, training uses Adam at learning rate 10^{-3} with batch size 256, and the shared weights are $\lambda_{\mathrm{adv},x} = \lambda_{\mathrm{adv},z} = 1$, $\lambda_{\mathrm{rec},x} = 3$, $\lambda_{\mathrm{rec},z} = 1$, an overall cycle weight of 5, a KL weight of 5×10^{-5}, and one discriminator update per generator update. The variance target is 0.01 with numerical stabilization 10^{-8}, and the joint-MMD and sliced-Wasserstein penalties are disabled. The remaining regularization weights are fixed per dataset to the values in Table 7. The EGM step is chosen by the pooled generation rank mentioned in S4, on validation samples only.

S3. Alternating latent and generator updates

Table 7: Regularization weights for the real datasets, fixed on validation data before any test point is scored.

		EGM			Iterative		
Dataset	z	α	λ_{var}	λ_{corr}	$\lambda_{\text{var}}^{\text{iter}}$	$\lambda_{\text{corr}}^{\text{iter}}$	λ_{PP}
BANK	8	0.1	0.01	0.3	0	0.3	1000
ParkTele	8	0	0	0.3	0	0.3	1000
Pendigits10	8	0	0	0.3	0.01	0.3	0
Vehicle	6	0	0	0	0	0	0
EEGEye	5	0.1	0.01	0	0.01	0	0

After EGM initialization, the sample-specific latent variables and generative parameters are refined by alternating stochastic updates. For a mini-batch $\mathcal{B}$, the latent representations are updated by minimizing the negative log posterior

$$\mathcal{J}_z = \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \left[-\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_i \mid \boldsymbol{z}_i) + \frac{1}{2} \|\boldsymbol{z}_i\|_2^2 \right]. \tag{31}$$

The latent variables are optimized using Adam with learning rate 0.005.

Conditional on the updated latent variables, the generator is optimized using

$$\mathcal{J}_{iter} = -\frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}_i \mid \boldsymbol{z}_i) + \lambda_{\text{PP}} \mathcal{L}_{\text{MMD,PP}} + \lambda_{\text{corr}}^{\text{iter}} \mathcal{L}_{\text{corr}} + \lambda_{\text{var}}^{\text{iter}} \mathcal{L}_{\text{logvar}}. \tag{32}$$

Here, $\mathcal{L}_{\text{MMD,PP}}$ is a prior-predictive multi-scale MMD penalty comparing the observed mini-batch with samples generated from fresh $\boldsymbol{z} \sim \mathcal{N}(0, I)$. Terms not enabled in a given configuration have zero weight. The generator is optimized using Adam with learning rate 0.005 and batch size 256.

For the real datasets, the iterative regularization weights are likewise fixed per dataset (Table 7). Every run starts from the EGM checkpoint frozen in the preceding stage, uses Adam at learning rate 0.005 with batch size 256, and is trained for $2{,}000$ epochs with a checkpoint every 50. The reported epoch is the one maximizing the validation decoder log-likelihood, frozen before any test point is scored.

S4. Step/Epoch selections

All architecture, regularization, EGM-step, and iterative-epoch decisions were made without access to the test observations. Selection used fixed validation observations and fixed random seeds.

For simulation experiments, both the EGM checkpoint and the iterative-updating checkpoint were selected with the same criterion, computed from held-out observations and generated samples. For a checkpoint grid $\mathcal{E}$ and the diagnostic set $\mathcal{M}$, we minimized the equal-weight average percentile rank

$$\widehat{e} = \arg\min_{e \in \mathcal{E}} \frac{1}{|\mathcal{M}|} \sum_{m \in \mathcal{M}} r_m(e), \qquad r_m(e) = \frac{\left|\{e' \in \mathcal{E} : d_m(e') < d_m(e)\}\right|}{|\mathcal{E}| - 1}, \tag{33}$$

where $d_m(e)$ is diagnostic m at checkpoint e, oriented so that smaller is better. The diagnostics are RBF MMD, mean symmetric KL, sliced Wasserstein distance, mean marginal Wasserstein distance, mean KS statistic, dimension-normalized correlation error, one minus the local inverse Simpson's index (LISI; Korsunsky et al., 2019), computed on the union of held-out and generated samples labeled by their source and rescaled to $[0, 1]$ so that 1 indicates perfect mixing, and one minus the fraction of adequately reproduced marginal standard deviations. This criterion requires neither ground-truth density nor known mixture centres. For the conditional datasets every diagnostic was computed per class and averaged with weights proportional to the observed class frequencies. The test split was evaluated only after all choices had been frozen. For Real UCI datasets, The EGM configuration and step are chosen by the validation generation-diagnostic rank sum; the iterative configuration and epoch by the validation decoder log-likelihood.

S5. Posterior simulation and bridge estimation

For each evaluation point, HMC was initialized near $E(\boldsymbol{x})$ using four chains, step size 0.003, and 10 leapfrog steps. The independent-GMM configuration retained 1600 posterior draws after 800 burn-in transitions. Half of the retained posterior draws were used to fit the proposal and the other half were reserved for bridge estimation. The involute and UCI evaluations used four HMC chains, an initial step size of 0.003, 10 leapfrog steps, 800 burn-in transitions per chain, and 1600 retained draws in total, divided equally between proposal fitting and bridge estimation. For MNIST, we used four chains with the same initial step size and number of leapfrog steps, 400 burn-in transitions per chain, 800 retained draws in total (400 for proposal fitting and 400 for bridge estimation), and 5000 proposal draws. In all experiments, chains were initialized from the encoder with independent Gaussian perturbations of standard deviation 0.01, and the step size was adapted during the first 80% of burn-in toward a target acceptance probability of 0.75.

The observation-specific proposal was constructed by fitting a five-component, full-covariance Gaussian mixture to the proposal-fitting HMC draws. Its fitted weights and means were retained, and its Gaussian kernels were replaced by Student-t kernels with three degrees of freedom. Let $\widehat{\boldsymbol{\Sigma}}_k^{\mathrm{GMM}}$ denote the covariance returned by the GMM fit. The GMM was fitted with diagonal covariance regularizer $\delta = 10^{-3}$. Before Cholesky factorization, the implementation further formed $\boldsymbol{L}_k\boldsymbol{L}_k^\top = \widehat{\boldsymbol{\Sigma}}_k^{\mathrm{GMM}} + \eta_k\boldsymbol{I}, \eta_k \geq \delta$, starting with $\eta_k = \delta$ and increasing it by factors of ten only if the factorization failed. Thus the scale used by the proposal includes both the GMM covariance regularization and the additional numerical jitter; it is not the unregularized within-component sample covariance.

The proposal was augmented by a standard-normal defensive component of weight 0.05, and 20,000 proposal samples were drawn. The bridge fixed point was initialized by the importance-sampling estimate and iterated on the log scale until numerical convergence, with tolerance 10^{-5} and at most 1000 iterations.

The bridge fractions were formed using a quantity (Vehtari et al., 2021) denoted by M_{eff}. In the implementation, draws from all HMC chains are pooled, randomly permuted, and divided into proposal-fitting and bridge subsets. The bridge subset is treated as a single sequence, an effective sample size is computed separately for each latent coordinate, and the minimum over coordinates is used as M_{eff}, truncated to the interval $[1, M_b]$. The resulting fractions are $s_p = \frac{M_{\mathrm{eff}}}{M_{\mathrm{eff}}+S}, s_q = \frac{S}{M_{\mathrm{eff}}+S}$.

Because the permutation does not preserve within-chain temporal order, this quantity should be understood as a heuristic weighting parameter computed from the pooled and shuffled bridge subset, rather than a conventional multi-chain Markov-chain ESS. All posterior kernels, proposal densities, importance weights, and bridge updates were evaluated on the log scale.

S6. Bridge Sampling Ablation and Running Time

Both estimators are evaluated on the same 500 held-out test points with M = 400 and the same defensive mixture proposal, so the training recipe is held fixed and the only difference is how the normalising constant $Z = p_{\boldsymbol{\theta}}(x_{\mathrm{obs}})$ is estimated from the posterior draws. Only the HMC budget M varies; the number of proposal draws is held at the production value $S = 20000$ throughout. Each point is the mean over three independent Monte Carlo replicates of the estimator, with the model, data, test points and budgets held identical across replicates. Beyond $M = 400$ the rank correlation of bridge sampling saturates, and the remaining differences are within about one standard error across replicates, as the estimate converges to the fitted model's density $p_{\hat{\theta}}$ rather than to the true density.

Panel (b) of Figure 5 turns to cost, reporting for each method the time to fit the model plus the time to score the 500 test points. The axis is logarithmic because the totals span two and a half orders of magnitude, from 2.1 minutes for CFMs to 756 minutes for Resflow. Testing is timed with the whole 500×17 test array handed to each method at once, on one card. The flows score it in a single forward pass and CFMs in a single solve of its density ODE. Roundtrip draws its $500 \times S$ importance samples as one array, and BayesNDE advances the 500×4 HMC chains in lockstep, returning a $500 \times M \times 6$ array of posterior draws, then carries its proposal draws and bridge iterations as $500 \times S$ arrays.

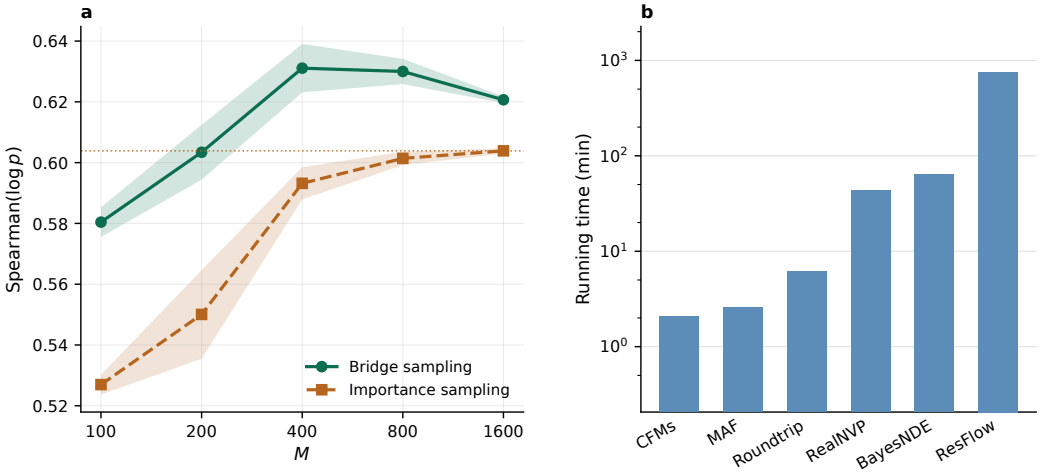


Figure 5: **(a)** Sampling-budget curve on the 15-dimensional independent GMM benchmark: rank correlation between $\log \hat{p}$ and the ground-truth log-density against the posterior budget M. Shaded bands are ± 1 standard error over three estimator replicates, and the dotted horizontal line marks the highest accuracy importance sampling attains at any budget, which bridge sampling already exceeds at $M = 400$. **(b)** Running time on BANK: the time to fit each estimator plus the time the fitted estimator takes to score 500 test points, drawn uniformly at random without replacement from the 4521-row test split. All six methods use all 17 columns and were timed on one node of the same type (Intel Xeon 6442Y, four cores, a single NVIDIA A40). The vertical axis is logarithmic.

Table 8: Sampling-budget sweep at $p = 15$, mean $\pm$ standard error over three estimator replicates on 500 test points. Δ is bridge minus importance sampling.

	Spearman($\log p$)		
M	Bridge	IS	Δ
100	$\mathbf{0.5804} \pm 0.0049$	0.5270 ± 0.0033	$+0.0535$
200	$\mathbf{0.6034} \pm 0.0090$	0.5501 ± 0.0146	$+0.0534$
400	$\mathbf{0.6311} \pm 0.0080$	0.5932 ± 0.0053	$+0.0379$
800	$\mathbf{0.6300} \pm 0.0042$	0.6014 ± 0.0022	$+0.0286$
1600	$\mathbf{0.6207} \pm 0.0011$	0.6039 ± 0.0009	$+0.0168$

The two Monte Carlo estimators each have a sampling budget to choose, and one rule picks it for both: the cheapest budget whose per-point estimates still rank the test set in agreement with that method's own converged estimate at Spearman ≥ 0.85. Roundtrip's rank agreement with its own converged estimate is 0.700 at $2\,500$ samples and 0.758 at $10\,000$, so it is timed at $40\,000$; BayesNDE reaches 0.850 at $M = 100$ and is timed there. At either budget testing is a small part of the total — 8.6 seconds for Roundtrip and 20.0 seconds for BayesNDE, against 6.0 and 63.8 minutes of training. The training part covers model fitting only. BayesNDE runs in 64.2 minutes in total. That is longer than CFMs, MAF and Roundtrip (2.1, 2.6 and 6.1 minutes) and about one and a half times RealNVP's 43.6 minutes, and twelve times shorter than Resflow's 756 minutes. On the same data, BayesNDE's mean test log-likelihood exceeds Roundtrip's by 5.7 nats.

A.5 Baseline methods

We compare BayesNDE against six density-estimation baselines spanning autoregressive flows, coupling flows, free-form residual flows, continuous-time flow matching, and the GAN-based estimator that is closest to our own setting. Every method is trained on exactly the same train/validation split, is evaluated on a byte-identical set of held-out points (we assert equality of the SHA-256 digests of the evaluation matrices across all methods). All baselines run their original authors' code, taken unmodified from the public releases; only the data interface and the input width are adapted.

Conditioning. BayesNDE fits a single model $p_{\boldsymbol{\theta}}(x \mid z, y)$ in which the one-hot label is concatenated to the input of every residual block of the generator, of the encoder, and of both discriminators; one set of parameters is shared across all labels and is fitted on the full training split. The baselines are unconditional density estimators whose public implementations have no label pathway, so for each of them we fit one independent unconditional expert per observed label, seeded at $42 + y$, and report the expert's density as $p(x \mid y)$. Neither construction assigns a density to the discrete label itself, and both are evaluated on the same points.

MADE. Masked Autoencoder for Distribution Estimation provides the autoregressive, single-pass reference point. We call the official `maf-master` entry point `experiments.train_made([100,100], 'relu', 'sequential')`, which builds `ml.models.mades.GaussianMade`: two hidden layers of 100 ReLU units, sequential input ordering, no batch normalization, and one Gaussian conditional per coordinate. Training uses the authors' `ml.trainers.SGD` with Adam at step size 10^{-3}, weight decay 10^{-6}, minibatch 100, validation monitored every epoch, and early stopping with patience 30.

MAF. Masked Autoregressive Flow stacks MADE blocks to obtain a deeper autoregressive density. We call `experiments.train_maf([100,100], 'relu', 10, 'sequential')`, giving `ml.models.mafs.MaskedAutoregressiveFlow` with 10 MADE blocks, each with two hidden layers of 100 ReLU units and sequential ordering, and batch normalization between consecutive blocks. The trainer is the same as for MADE with the authors' flow step size 10^{-4}.

Real NVP. Real NVP is the affine-coupling counterpart, whose Jacobian is triangular by construction rather than by masking order. We call `experiments.train_Real NVP([100,100], 'tanh', 'relu', 10)`, giving `ml.models.nvps.Real NVP` with 10 coupling layers, scale and translation networks of two hidden layers of 100 units with `tanh` and ReLU activations respectively, and batch normalization. Optimisation settings match MAF.

CFMs. Conditional Flow Matching (CFMs) learns a continuous-time velocity field by regression against a conditional probability path, avoiding ODE simulation during training (Lipman et al., 2023; Tong et al., 2024). We use the TorchCFM implementation from the official `conditional-flow-matching` repository of Tong et al. (2024), including its `CFMLitModule`, conditional flow-matching objective, and `VelocityNet` implementation.

Resflow. Residual Flows give an invertible free-form architecture with an unbiased log-determinant estimator, and are the strongest baseline we find in high dimension. We use the official `residual-flows` layers with 100 `iResBlocks`. Each block is a residual network of Swish

activations and spectrally normalised linear layers with Lipschitz coefficient 0.9, 5 power iterations, and domain/codomain exponents 2.0; following the authors' `train_toy.py` condition literally, the final layer is zero-initialised only when its output width is 2. The log-determinant uses the unbiased geometric Russian-roulette series (`n_dist='geometric'`, one sample, no exact trace). We train with Adam at learning rate 10^{-3} and weight decay 10^{-5}, batch size 500, for $50{,}000$ iterations, refreshing the Lipschitz constants every 5 training iterations and with 200 iterations before each evaluation, validating every 100 iterations and retaining the best-validation checkpoint.

Roundtrip. Roundtrip is the closest comparator to our setting: it places a latent variable under a generative network and estimates pointwise density by one-sided importance sampling. We use the authors' original TensorFlow density-estimation implementation and architecture: a generator with 10 hidden layers of 512 units, an encoder with 10 hidden layers of 256 units, a latent discriminator with two hidden layers of 128 units, and a data discriminator with four hidden layers of 256 units. Training uses Adam with learning rate 2×10^{-4}, batch size 64, cycle weights $\alpha = \beta = 10$, and an image pool of size 50. Models are trained for at most 100 epochs, with validation-based model selection beginning at epoch 30 and early-stopping patience 5. Pointwise densities are estimated using the authors' one-sided importance sampler with a degree-one Student-t proposal, $40{,}000$ importance samples. Selections of standard deviation and proposal scale follows the original paper.

Execution environments. MADE, MAF and Real NVP are the original Theano 1.0.5 implementations and are executed in a dedicated Python 3.10 environment through a subprocess worker; CFMs and Resflow run under PyTorch 1.12 on a single GPU; Roundtrip and BayesNDE run under the project TensorFlow environment. Because the three environments cannot share a process, we verified that the evaluation matrices they read are byte-identical by comparing SHA-256 digests computed by independent, separately audited implementations of the same hashing routine.