

---

# Variational Pólya Tree

---

**Lu Xu**<sup>\*</sup>

The University of Hong Kong  
xulu0908@hku.hk

**Tsai Hor Chan**<sup>\*</sup>

University of Pennsylvania  
Tsaihor.Chan@PennMedicine.upenn.edu

**Kwok Fai Lam**

Hong Kong Metropolitan University  
kflam@hkmu.edu.hk

**Lequan Yu**

The University of Hong Kong  
lqyu@hku.hk

**Guosheng Yin**<sup>†</sup>

The University of Hong Kong  
gyin@hku.hk

## Abstract

Density estimation is essential for generative modeling, particularly with the rise of modern neural networks. While existing methods capture complex data distributions, they often lack interpretability and uncertainty quantification. Bayesian nonparametric methods, especially the Pólya tree, offer a robust framework that addresses these issues by accurately capturing function behavior over small intervals. Traditional techniques like Markov chain Monte Carlo (MCMC) face high computational complexity and scalability limitations, hindering the use of Bayesian nonparametric methods in deep learning. To tackle this, we introduce the variational Pólya tree (VPT) model, which employs stochastic variational inference to compute posterior distributions. This model provides a flexible, nonparametric Bayesian prior that captures latent densities and works well with stochastic gradient optimization. We also leverage the joint distribution likelihood for a more precise variational posterior approximation than traditional mean-field methods. We evaluate the model performance on both real data and images, and demonstrate its competitiveness with other state-of-the-art deep density estimation methods. We also explore its ability in enhancing interpretability and uncertainty quantification. Code is available at <https://github.com/howardchanth/var-polya-tree>.

## 1 Introduction

Density estimation has gained significant importance in generative modeling, particularly with the advent of modern neural networks [3, 28], such as normalizing flows and autoregressive networks. These techniques have found numerous applications, including image generation and large language modeling, highlighting their versatility and effectiveness in capturing complex data distributions. While many recent efforts focus on enhancing feature learning and transforming densities [46, 9], these deep density estimation networks often struggle with interpretability, raising concerns about their trustworthiness [30].

---

<sup>\*</sup>Equal contributions

<sup>†</sup>Corresponding author

Bayesian nonparametric methods provide an appealing alternative, enabling flexible modeling that naturally adapts complexity according to data and rigorously quantifies uncertainty [32]. Incorporating Bayesian nonparametric priors such as Dirichlet processes (DP) into deep neural architectures has attracted substantial attention, yielding notable progress in regression [42] and clustering [26]. However, DP-based approaches inherently induce discreteness, making them suboptimal for directly modeling continuous distributions.

In contrast, the Pólya tree (PT) is specifically suited to continuous distributions, producing probability measures absolutely continuous with respect to the Lebesgue measure [35, 4, 29]. Yet, despite their appealing theoretical properties, PT priors have rarely been combined with modern deep learning frameworks due to computational challenges. Traditional posterior inference via Markov chain Monte Carlo (MCMC) suffers scalability limitations, becoming prohibitively expensive as data scales [48, 4].

To bridge this gap, we propose the variational Pólya tree (VPT), the first variational Bayesian integration of PT priors with deep neural networks. Our VPT leverages stochastic variational inference, enabling scalable training via stochastic gradient optimization. Importantly, our method does not rely on the simplifying independence assumptions typical of mean-field approximations. Instead, we exploit the PT prior’s intrinsic hierarchical structure and conjugacy properties, allowing a tractable and exact joint posterior over Beta-distributed node weights. This approach maintains the rich dependency structure across tree nodes, facilitating efficient and precise backpropagation.

We validate VPT across diverse tasks, including high-dimensional tabular data and image density estimation using normalizing flow architectures. Unlike existing deep density estimation works that mainly focus on transformation and feature learning architectures [18, 6, 39, 15], we present a novel prior and demonstrate how VPT can enhance interpretability and provide a measure of uncertainty.

Our contributions can be summarized as follows: (1) We introduce the first variational inference framework that makes PT priors practically useful and can be used as plug-and-play components for continuous density estimation and deep generative modeling, integrating seamlessly with modern neural architectures, e.g., flows, variational autoencoders (VAEs). (2) We leverage the joint posterior likelihood as the variational objective to provide an exact approximation of the posterior distribution of the PT while minimizing computational complexity. (3) Empirical evaluations on various datasets demonstrate that our VPT prior can achieve superior performance in density estimation compared to existing methods. Additionally, ablation analysis reveals that a deeper tree generally leads to more accurate density estimation. The framework incurs minimal overhead—less than 0.05% additional memory and at most  $1.3 \times$  runtime—while remaining end-to-end trainable with standard deep learning architectures. (4) We demonstrate that with the VPT prior, the model can effectively learn both the estimated density distributions and the associated uncertainties. Our findings also indicate that the latent features acquired using the VPT prior are significantly more meaningful than those derived from traditional distributions.

## 2 Related Works

### 2.1 Deep Density Estimation

Deep generative models have achieved significant success in density estimation, especially for high-dimensional data like images [3, 28]. These models typically transform latent variables, sampled from known distributions, into data points, enabling flexible and expressive density modeling.

Normalizing flows use sequences of invertible transformations, parameterized by neural networks, to model complex distributions. They enable efficient likelihood evaluation by maintaining tractable Jacobian computations. For example, NICE [8] uses coupling layers, while RealNVP [9] extends this with additive and multiplicative coupling layers for improved expressiveness. Autoregressive methods decompose a  $D$ -dimensional density into one-dimensional conditional densities, functioning as a specialized form of normalizing flow. MAF [39] uses masked neural networks to model these conditional distributions, while IAF [23] enables more efficient sampling through inverse transformations. VAEs [20] introduce stochastic transformations to learn compact representations but often suffer from posterior collapse due to the evidence lower bound (ELBO). Diffusion models [17], a hierarchical extension of VAEs, use forward noise addition and reverse denoising processes to generate data.

Despite these advances, most deep density estimators still rely on parametric assumptions and lack a principled way to encode interpretability and uncertainty about the distribution. Most recent works in this area focus on learning powerful and tractable transformations from input to the latent density. For example, Glow [22] introduces invertible  $1 \times 1$  convolutions to improve expressiveness. Neural autoregressive flow (NAF) [18] employs autoregressive models within a normalizing flow framework, allowing for efficient and expressive density estimation. MuLAN [41] utilizes adaptive noise techniques within diffusion models to improve sample quality and diversity. We propose a Bayesian nonparametric prior that can be compatible with these transformation approaches, which is not only flexible to train but also provides interpretability and uncertainty quantification naturally.

## 2.2 Bayesian Nonparametric Methods

Bayesian nonparametric (BNP) models provide a framework for adapting model complexity to data [47, 40, 27, 16, 19, 43]. Classic examples include Dirichlet processes (DPs), which enable unbounded mixture components, and have been extended to deep learning, such as stick-breaking VAEs and Beta-Bernoulli processes for infinite latent features [33, 10]. However, DPs concentrate probability on discrete distributions, making them unsuitable for continuous densities without additional smoothing [35]. Variational inference for DPs also requires truncating the infinite process, which can result in biased estimates.

Pólya trees, in contrast, naturally model continuous densities by recursively partitioning the domain with random probabilities. With appropriate hyper-parameters, they place probability 1 on the space of continuous distributions [25]. Unlike DPs, PTs avoid discreteness and external smoothing. Despite this advantage, their use in modern machine learning has been limited, as existing applications relied on computationally expensive methods like MCMC [2]. Crucially, no previous work has incorporated PT priors into deep neural architectures with variational Bayes.

We address this gap by introducing a PT-based variational inference framework for deep generative models. Our proposed framework is not a plug-and-play prior placement but a principled, nontrivial integration that bridges classical Bayesian nonparametrics with modern deep learning. It leverages the distinctive theoretical properties of PTs, enabling benefits such as robustness, interpretability, generalization, and calibrated uncertainty estimation. PTs’ conjugacy allows efficient updates of branch probabilities via Beta–Binomial computation, enabling a structured variational posterior that retains dependencies across the tree. Unlike mean-field methods, which assume independence [45, 11, 14], our approach preserves hierarchical coherence, avoiding oversimplified approximations.

By integrating the flexibility of BNP priors with the scalability of deep learning, our method enables efficient end-to-end training of continuous density estimators. This approach overcomes the limitations of DP-based priors for continuous data and advances adaptive, interpretable density modeling in deep generative frameworks.

## 2.3 Tree-based Models

The use of binary trees is prevalent in machine learning [7, 42, 1]. In addition to well-known methods like decision trees, random forests, boosting, and XGBoost [5], Bayesian approaches to tree models have gained popularity, particularly with advancements in computational capacity.

Further, Bayesian inference has gained enormous popularity for modeling the distribution of trees, as seen in Bayesian additive regression trees (BART) [1]. Salazar [42] introduces a variational tree model for regression that establishes a Bayesian nonparametric prior on the tree space. However, like traditional methods, it partitions the sample space rather than the probability space.

In contrast, our approach uses a PT prior to model continuous densities. A tree space that partitions the probability space would offer greater modeling flexibility, accommodating a wider range of probability measures and enabling various tasks beyond regression. Our variational algorithm preserves dependencies across tree levels, ensuring coherent uncertainty propagation and avoiding the limitations of the mean-field approximation.

### 3 Methodology

#### 3.1 The Pólya Tree Prior

The PT prior is a Bayesian nonparametric approach for constructing a random probability measure. The construction is based on a recursive partition of the domain, which can be viewed as a random histogram with the sizes of the bins made sequentially smaller [38].

Consider the bins of a histogram defined by a partition of the sample space into nonempty subsets  $\{B_{\epsilon_{1:L}}\}$ , with subsets indexed by an  $L$ -digit binary number  $\epsilon_{1:L} = \epsilon_1 \cdots \epsilon_L$ , where each element  $\epsilon_j$  ( $j = 1, \dots, L$ ) takes values of 0 or 1. At level  $j$ , we define random probabilities  $P(B_{\epsilon_{1:j}})$  for each bin. Consider refining the partition by splitting each bin into *left* and *right* parts,  $B_{\epsilon_{1:j}} = B_{\epsilon_{1:j}0} \cup B_{\epsilon_{1:j}1}$  and define random probabilities for the refined histogram by conditional probabilities  $Y_{\epsilon_{1:j}0} = P(B_{\epsilon_{1:j}0}|B_{\epsilon_{1:j}})$  and  $Y_{\epsilon_{1:j}1} = P(B_{\epsilon_{1:j}1}|B_{\epsilon_{1:j}})$ . For any integer  $L \geq 0$ , this recursive refinement of bins defines a sequence of nested partitions.

A random probability measure  $\mathbb{P}$  follows a PT distribution  $\text{PT}(\mathcal{A}, \{\beta_j\}_{j=1}^L)$  with parameters  $\mathcal{A} = \{\alpha_\epsilon\}$  on the sequence of partitions  $\{\beta_j\}_{j=1}^L$  where  $\alpha_\epsilon$  refers to the collection of Beta distribution random variables (i.e., the branching probabilities), if there exist splitting probabilities  $0 \leq Y_\epsilon \leq 1$  such that for any  $\epsilon$ : (1) The variables  $Y_{\epsilon_0}$  are mutually independent and the posterior follow a  $\text{Beta}(\alpha_{\epsilon_0}, \alpha_{\epsilon_1})$  distribution; (2)  $Y_{\epsilon_1} = 1 - Y_{\epsilon_0}$ ; (3) For any  $L \geq 0$ ,  $P(B_{\epsilon_{1:L}}) = \prod_{j=1}^L Y_{\epsilon_1 \dots \epsilon_j}$ .

Figure 1 illustrates this recursive representation. In summary, the PT prior defines a random probability distribution on distributions on  $[0, 1]$  by assigning any partitioning subset  $B_\epsilon$  with probability  $P(B_\epsilon) = \prod_{j=1, \epsilon_j=0}^L Y_{\epsilon_1 \dots \epsilon_{j-1}0} \times \prod_{j=1, \epsilon_j=1}^L (1 - Y_{\epsilon_1 \dots \epsilon_{j-1}0})$ . Algorithm 1 shows the overall workflow of our VPT.

#### 3.2 Variational Pólya Tree

**Variational Objective.** Our VPT approach leverages the intrinsic hierarchical structure of the Pólya tree to avoid traditional independence assumptions common in mean-field variational inference. Specifically, we parameterize the tree partitions  $\{\beta_j\}_{j=1}^L$  by learnable neural networks. Due to the conjugacy of the PT prior, the posterior retains the PT form, allowing closed-form updates for each node’s Beta parameters. This structural property preserves hierarchical dependencies among latent variables without any simplifying assumptions, which distinguishes VPT from the classical mean-field approximation.

Let  $\mathbf{x} = \{x_i\}_{i=1}^N$  be the observed data,  $\{\beta_{i,j}\}_{j=1}^L$  be the  $L$ -level tree partition probabilities corresponding to the  $i$ -th observation, and  $\mathcal{Y}^L$  be the set of all  $Y_{\epsilon_{1:j}}$  in the tree. Following

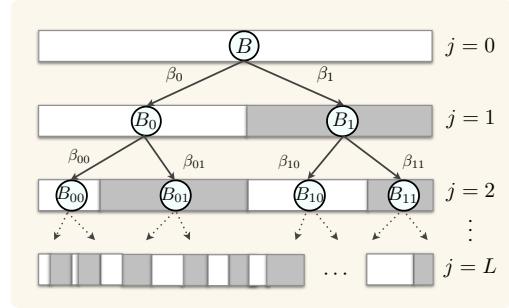


Figure 1: Graphical illustration of the Pólya tree construction. The PT prior randomly splits  $B_{\epsilon_{1:j}}$  into two subintervals  $B_{\epsilon_{1:j}0}$  and  $B_{\epsilon_{1:j}1}$  with Beta-distributed probabilities.

---

#### Algorithm 1 Training Procedure of Variational Pólya Tree (VPT)

---

```

1: Input: Data points  $\{x_i\}_{i=1}^N$ , dimension  $D$ , tree level  $L$ , number of epochs  $n_{\text{epoch}}$ , learning rate  $\eta$ .
2: Compute total number of nodes per dimension:  $n_{\text{nodes}} = 2^L - 1$ 
3: Initialize Beta parameters for all nodes:  $\alpha_{\epsilon_{1:j-1}0} = 1, \alpha_{\epsilon_{1:j-1}1} = 1$ 
4: for epoch = 1, ...,  $n_{\text{epoch}}$  do
5:   for each dimension  $d = 1, \dots, D$  do
6:     for each node  $j = 1, \dots, n_{\text{nodes}}$  do
7:       Sample split:
8:        $Y_{\epsilon_{1:j-1}0}^{(d)} | \mathbf{x} \sim \text{Beta}(\alpha_{\epsilon_{1:j-1}0}^{(d)}, \alpha_{\epsilon_{1:j-1}1}^{(d)})$ 
9:       Compute partition intervals  $B_{\epsilon_{1:j}}^{(d)}$ .
10:      end for
11:    end for
12:    Compute joint posterior in Eq. (1),
13:     $p(\{\beta_j\}_{j=1}^L, \mathcal{Y}^L | \mathbf{x})$ , and
14:     $\mathcal{L}_{\text{VPT}} = \log p(\{\beta_j\}_{j=1}^L, \mathcal{Y}^L | \mathbf{x})$ 
15:    Update  $\alpha_{\epsilon_{1:j-1}0}, \alpha_{\epsilon_{1:j-1}1}$ .
16:  end for

```

---

Paddock et al. [38], the joint posterior distribution of an  $L$ -level tree given  $\mathbf{x}$  can be written as

$$p(\{\beta_j\}_{j=1}^L, \mathcal{Y}^L | \mathbf{x}) = \prod_{i=1}^N \frac{1}{\nu(B_{\epsilon_{i,1:L}})} \prod_{j=1}^L Y_{\epsilon_{1:j}} p(\beta_{i,j}) \prod_{\forall \{\epsilon_{1:j-1}\}} Y_{\epsilon_{1:j-1}0}^{(\alpha_{\epsilon_{1:j-1}0} - 1)} Y_{\epsilon_{1:j-1}1}^{(\alpha_{\epsilon_{1:j-1}1} - 1)}, \quad (1)$$

where  $\nu(B_{\epsilon_{i,1:L}}) = \prod_{j=1}^L \beta_{i,j}^{1-\epsilon_{i,j}} (1-\beta_{i,j})^{\epsilon_{i,j}}$  and  $p(\beta_{i,j})$  is the prior distribution for  $\beta_{i,j}$ . Data points  $x_i$  influence densities through two components, the probabilities  $B_{\epsilon_{1:j}}$  indicating the likelihood of  $x_i$  falling into a specific subset, and the conditional probabilities  $Y_{\epsilon_{1:j}}$  that govern the likelihood of  $x_i$  in child subsets given its parent subset. The gradients can be computed and backpropagated to update the parameters  $\alpha_{\epsilon_{1:j-1}0}$  and  $\alpha_{\epsilon_{1:j-1}1}$  of the Beta distributions within the Pólya tree.

We optimize VPT parameters by maximizing the log-likelihood (i.e., the ELBO)  $\mathcal{L}_{\text{VPT}} = \log p(\mathbf{x} | \{\beta_j\}_{j=1}^L, \mathcal{Y}^L)$ . Note that our method does not explicitly include a Kullback–Leibler (KL) divergence term, and the corresponding entropy of the posterior acts as an implicit regularization. This entropy term, which appears in the ELBO, plays a similar role by controlling the spread of the variational distribution and balancing data fit versus uncertainty. Nevertheless, incorporating an explicit KL (as shown in Appendix A) or entropy term can further balance exploration of the prior.

**Density Estimation with Flow-based Models.** The VPT prior alone defines a flexible density over a sample space. For complex high-dimensional data, we can further enhance its flexibility by embedding the VPT with deep neural networks. Taking flow-based networks as an example, we use the PT density as the base distribution and apply an invertible deep transformation  $f$  from the latent space to the observed data space. Formally, let  $\mathbf{x}$  represent samples drawn from an unknown  $D$ -dimensional distribution. The goal of density estimation is to model the underlying distribution function  $F(\cdot)$ . Instead of assuming a fixed parametric form for  $F(\cdot)$  as in the previous work [42], we follow a Bayesian nonparametric approach by placing a VPT prior on  $F(\cdot)$ , thereby treating the distribution itself as random. Consider a bijective function  $f : \mathcal{X} \rightarrow \mathcal{Z}$  with the inverse function  $f^{-1}$  and Jacobian  $J_f(x)$ . Under this transformation, the density in the latent space  $\mathcal{Z}$  can be computed as

$$p_Z(z) = p_X(f^{-1}(z)) |\det(J_f(f^{-1}(z)))|,$$

where  $p_X(x)$  is modeled by VPT prior. The determinant of the Jacobian ensures the validity of the density transformation, which can be computed following Dinh et al. [8]. The model is optimized by maximizing the log-likelihood:  $\mathcal{L} = \log p_Z(z)$ . Unlike the standard flows with fixed base distributions, VPT base distribution is learned and updated via the posterior of the PT. In practice, we alternate analytic posterior updates of the PT parameters with gradient-based optimization.

### 3.3 Computation of Intervals

We use a tree data structure to compute the intervals of each leaf node [38]. For the  $i$ -th sample  $x_i$ , we first sample  $\{\beta_{i,j}\}_{j=1}^L$  from its variational posterior distribution. For dimension  $d$  where  $x_i^{(d)} \in \mathbb{R}$ , the interval  $(0, 1]$  is split into two subintervals with probabilities  $\beta_0^{(d)}$  and  $1 - \beta_0^{(d)}$ , so that the subinterval  $B_0^{(d)}$  is of length  $\beta_1^{(d)}$  and  $B_1^{(d)}$  is of length  $1 - \beta_1^{(d)}$ . Next, each of  $B_0^{(d)}$  and  $B_1^{(d)}$  is further split into two subintervals where the splitting probability depends on  $\beta_2^{(d)}$  as follows,

$$\begin{aligned} B_{00}^{(d)} &= (0, \beta_1^{(d)} \beta_2^{(d)}), \quad B_{01}^{(d)} = (\beta_1^{(d)} \beta_2^{(d)}, \beta_1^{(d)}) \\ B_{10}^{(d)} &= (\beta_1^{(d)}, \beta_1^{(d)} + (1 - \beta_1^{(d)}) \beta_2^{(d)}), \quad B_{11}^{(d)} = (\beta_1^{(d)} + (1 - \beta_1^{(d)}) \beta_2^{(d)}, 1]. \end{aligned}$$

The intervals at level  $j$  are obtained iteratively by applying the above rule.

### 3.4 Posterior Inference and Interpretability

With the learned  $\{\beta_{i,j}\}_{j=1}^L$  and  $\{\alpha_j\}_{j=1}^L$  for the  $i$ -th sample  $x_i$  with  $D$  dimensions, we can sample from the Beta distribution to obtain the probability of splits. Due to the conjugacy of the PT prior, the posterior predictive distribution has an intuitive form,

$$p(x | \mathcal{Y}^L) = \frac{1}{\nu(B_L(x))} \prod_{d=1}^D \prod_{j=1}^L \left( Y_{s_j^{(d)}(x)0}^{(d)} \right)^{1-\epsilon_j^{(d)}(x)} \left( 1 - Y_{s_j^{(d)}(x)0}^{(d)} \right)^{\epsilon_j^{(d)}(x)}.$$

where  $\mathcal{Y}^L = \{Y_{\epsilon_{1:j-1}0}^{(d)}\}$ ,  $Y_{\epsilon_{1:j-1}0}^{(d)}$  is the (left) split variable at node  $\epsilon_{1:j-1}$  in dimension  $d$ , and under the variational posterior  $Y_{\epsilon_{1:j-1}0}^{(d)} | \mathbf{x} \sim \text{Beta}(\alpha_{\epsilon_{1:j-1}0}^{(d)}, \alpha_{\epsilon_{1:j-1}1}^{(d)})$  with learned shape parameters  $\alpha_{\epsilon_{1:j-1}0}^{(d)}, \alpha_{\epsilon_{1:j-1}1}^{(d)}$ ,  $s_j^{(d)}(x) = \epsilon_{1:j-1}^{(d)}(x)$  is the parent index used at level  $j$ ,  $B_L(x) = \prod_{d=1}^D B_{\epsilon_{1:L}^{(d)}(x)}^{(d)}$  is the depth- $L$  leaf containing  $x$ , and  $\nu(B_L(x)) = \prod_{d=1}^D |B_{\epsilon_{1:L}^{(d)}(x)}^{(d)}|$  is its volume.

This hierarchical organization of posterior parameters makes the model highly interpretable. One can traverse the tree to see how the model allocates probability at different scales and locations. Each level of the tree gives a coarse-to-fine view of the density.

**Uncertainty quantification.** As a Bayesian nonparametric method, one of the key advantages of our VPT is to learn a distribution over the density estimation. This allows us to obtain the variance of the learned model. Given a trained  $L$ -level VPT, the posterior variance can be modeled as the mean over the variances of terminal node distributions, where the posterior variance of each subinterval can be computed easily from the Beta distribution.

## 4 Experiments

We evaluate VPT on multiple density estimation tasks. We first demonstrate the qualitative benefits of VPT compared to traditional Gaussian and logistic priors using simple synthetic data. We then assess VPT quantitatively on UCI and image datasets. Our findings indicate that models with VPT priors not only achieve superior likelihood results but also provide robust uncertainty estimates.

**Implementation details.** Training a VPT involves optimizing a set of Beta distribution parameters. To ensure the Beta distribution parameters remain positive, we reparameterize them via a softplus( $\cdot$ ) transformation. To integrate VPT prior seamlessly with standard generative networks, we use a sigmoid layer to project latent variables into the interval  $[0, 1]$ . This sigmoid mapping is computationally efficient, adding minimal complexity since its Jacobian determinant can be computed trivially.

### 4.1 Density Estimation with 2D Synthetic Data

We first illustrate the capability of our VPT prior using three common synthetic datasets, a ring of 8 Gaussians, two interwoven spirals, and a checkerboard pattern. For these experiments, we adopt a simple block neural autoregressive flow (Block-NAF) architecture as the feature learning backbone, specifically employing one flow layer consisting of two hidden layers, each with 50 units. This modest architecture choice is intentional, as our primary goal is not to achieve state-of-the-art performance on synthetic benchmarks, but rather to clearly demonstrate the benefit of the VPT prior compared to standard priors (Gaussian and logistic) under identical model complexity.

In Figure 2, we compare the learned densities using three different priors, a 2-level VPT, a 3-level VPT, and an isotropic Gaussian. While the isotropic Gaussian prior allows the model to approximate the general shape of multimodal distributions, it struggles to capture sharp boundaries and disconnected regions accurately. In contrast, both the 2-level and 3-level VPT priors clearly yield more precise representations, effectively capturing regions of low density and the multimodal structure inherent in the data. Moreover, increasing the level of the VPT prior from two to three levels further enhances its modeling capacity, resulting in notably improved density estimation and clearer separation among data clusters.

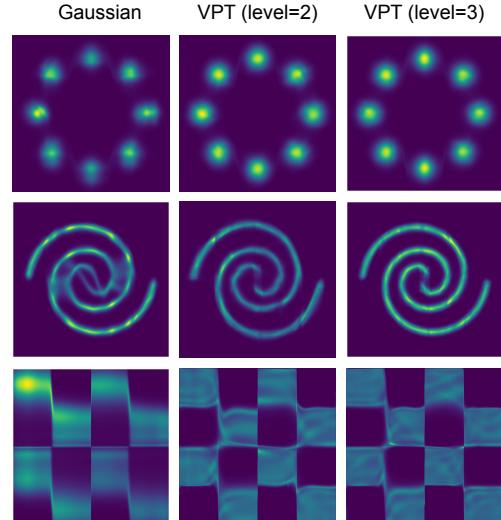


Figure 2: Density estimation results of an isotropic Gaussian prior, a 2-level VPT, and a 3-level VPT with 2D synthetic datasets.

Table 1: Log-likelihoods (standard deviations) on the test sets for density estimation using our VPT and baseline methods on real datasets with 5 runs. Higher values indicate better estimation.

Methods	POWER	GAS	HEPMASS	MINIBOONE	BSDS300
Real NVP [9]	0.17 <sub>(0.01)</sub>	8.33 <sub>(0.14)</sub>	-18.71 <sub>(0.02)</sub>	-13.55 <sub>(0.49)</sub>	153.28 <sub>(1.78)</sub>
Glow [22]	0.17 <sub>(0.01)</sub>	8.15 <sub>(0.40)</sub>	-18.92 <sub>(0.08)</sub>	-11.35 <sub>(0.07)</sub>	155.07 <sub>(0.03)</sub>
MADE MoG [39]	0.40 <sub>(0.01)</sub>	8.47 <sub>(0.02)</sub>	-15.15 <sub>(0.02)</sub>	-12.27 <sub>(0.47)</sub>	153.71 <sub>(0.28)</sub>
FFJORD [15]	0.46 <sub>(0.01)</sub>	8.59 <sub>(0.12)</sub>	-14.92 <sub>(0.08)</sub>	-10.43 <sub>(0.04)</sub>	157.40 <sub>(1.78)</sub>
MAF MoG [39]	0.30 <sub>(0.01)</sub>	9.59 <sub>(0.02)</sub>	-17.39 <sub>(0.02)</sub>	-11.68 <sub>(0.44)</sub>	156.36 <sub>(0.28)</sub>
TAN [36]	0.60 <sub>(0.01)</sub>	12.06 <sub>(0.02)</sub>	<b>-13.78</b> <sub>(0.02)</sub>	-11.01 <sub>(0.48)</sub>	159.80 <sub>(0.07)</sub>
NAF-DDSF [18]	0.62 <sub>(0.01)</sub>	11.96 <sub>(0.33)</sub>	-15.09 <sub>(0.40)</sub>	-8.86 <sub>(0.15)</sub>	157.43 <sub>(0.30)</sub>
Block-NAF [6]	0.57 <sub>(0.01)</sub>	11.01 <sub>(0.10)</sub>	-15.06 <sub>(0.08)</sub>	-8.90 <sub>(0.31)</sub>	156.33 <sub>(0.81)</sub>
VPT ( $L = 4$ )	<b>0.67</b> <sub>(0.01)</sub>	11.92 <sub>(0.10)</sub>	-15.01 <sub>(0.04)</sub>	-8.71 <sub>(0.25)</sub>	158.59 <sub>(1.02)</sub>
VPT ( $L = 6$ )	0.61 <sub>(0.01)</sub>	<b>12.20</b> <sub>(0.06)</sub>	-13.94 <sub>(0.05)</sub>	<b>-8.51</b> <sub>(0.03)</sub>	<b>162.72</b> <sub>(0.94)</sub>

## 4.2 Density Estimation with Real Data

We perform density estimation on five tabular UCI datasets, POWER, GAS, HEPMASS, MINIBOONE, and BSDS300. Detailed information about these datasets is summarized in Appendix B (Table A1). We follow the preprocessing procedure outlined in [39]. For this experiment, we employ a Block-NAF [6] as our feature learning architecture, which has fewer parameters compared to the original neural autoregressive flow [18]. Consistent with the Block-NAF methodology, we train 5 stacked flows, each with 2 layers and  $20D$  hidden units, where  $D$  represents the input dimension. The model is trained using the Adam optimizer, with a learning rate of  $10^{-2}$  for the Block-NAF flow and 0.1 for the variational Pólya tree.

**Results.** Table 1 summarizes the reported log-likelihood values. With a 4-level VPT, our model outperforms Block-NAF across all datasets. Additionally, with a 6-level VPT, our model demonstrates an even greater margin of improvement over Block-NAF, except on the POWER dataset.

By comparing our results with deeper density estimation models, we observe that our VPT models achieve the state-of-the-art performance across various datasets, with the exception of the HEPMASS dataset, where we attain results comparable to TAN [36].

**Number of parameters.** We compare model complexity against NAF [18] and Block-NAF [6]. Our VPT adds only  $(2^L - 1) \times 2 \times D$  parameters to Block-NAF, a negligible overhead compared to NAF. Despite this minimal increase, VPT significantly surpasses Block-NAF in performance, demonstrating superior modeling power and efficiency (see Table 2).

Table 2: Number of model parameters, relative to Block-NAF. Block-NAF and NAF use Gaussian priors, while VPTs with  $L = 4$  or  $6$  use the Block-NAF backbone with the VPT prior. The notation “ $\times$ ” indicates a multiplicative factor, while “ $+$ ” indicates an additive increase.

Datasets	Block-NAF	NAF	VPT(4)	VPT(6)
POWER	414, 213	$\times 4.57$	+180	+756
GAS	401, 741	$\times 2.60$	+240	+1008
HEPMASS	9, 272, 743	$\times 35.88$	+630	+2646
MINIBOONE	7, 487, 321	$\times 87.91$	+1290	+5418
BSD300	36, 759, 591	$\times 16.48$	+1890	+7938

## 4.3 Image Density Estimation and Image Generation

We further test our method on two image datasets: MNIST and CIFAR-10 [24]. We employ a classic flow-based network NICE [8] as our feature learning backbone, and we use the same settings as in the original paper. Additional details of implementations are presented in the Appendix.

**Results.** In Figure 3, we compare our VPT method with the vanilla NICE method [8] using two priors, Gaussian and logistic distributions. It is important to note that the metric employed in the original paper is log-likelihoods in the logit space, which is not directly comparable across different

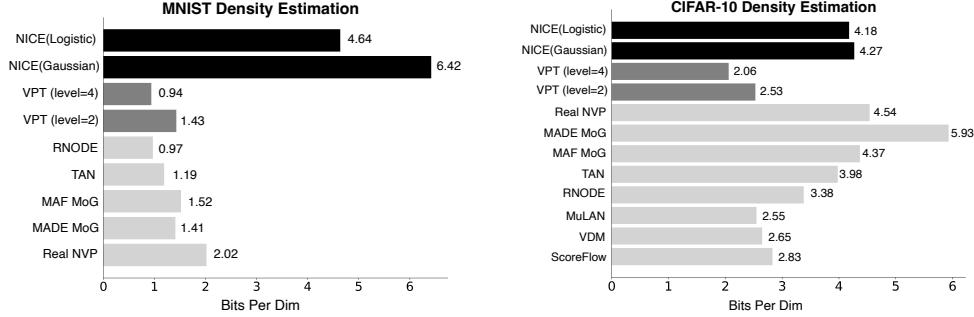


Figure 3: Negative log-likelihood in bits-per-dimension on the MNIST and CIFAR-10 test sets. Our method, VPT (shown in gray), shares the same architecture as NICE (shown in black). The results from other methods (shown in light gray) are sourced from their respective papers.

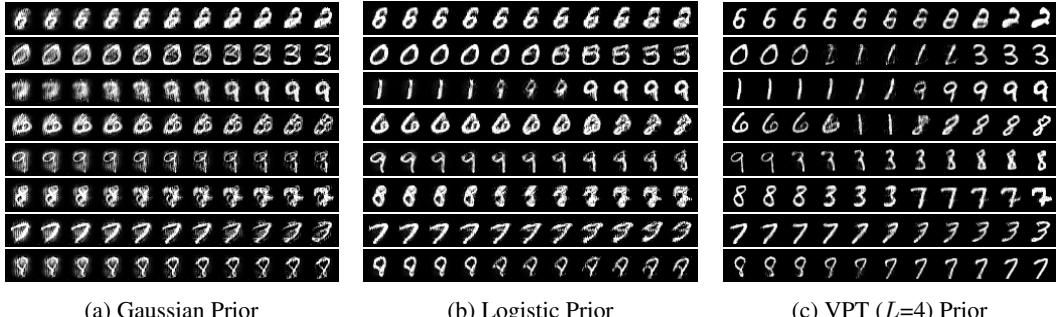


Figure 4: Interpolation on the MNIST images. While with the Gaussian prior and logistic prior the latent spaces are focused on the pixel-level features, the mixed latent features with our VPT prior reveal how the particular tree nodes are structured.

estimation methods. Therefore, we re-implemented their experiments and report the normalized negative log-likelihood in bits-per-dimension [39]:  $-\frac{1}{D} \log_2 p(x)$ . Our results indicate that the VPT prior significantly enhances the likelihood for both datasets.

We also compare our results with a variety of density estimation methods in Figure 3. These include flow-based methods such as Real NVP [9], MADE MoG [39], and MAF MoG [39]; RNN-based transformation approaches like TAN [36]; the Neural ODE method (RNODE) [12]; and more recently developed diffusion-based methods, including ScoreFlow [44], VDM [21], and MuLAN [41]. We observe that our VPT prior performs best among all the flow-based density estimation methods, while demonstrating comparable performance with the recently developed diffusion-based methods. With the 4-level VPT, our VPT achieves a negative log-likelihood of 0.94 on MNIST, which is similar to 0.97 by RNODE. On the CIFAR-10 data, our results with the 2-level VPT are comparable to those of MuLAN, while the performance can be further enhanced with the 4-level VPT.

#### 4.4 Uncertainty Analysis and Interpretation

**Uncertainty analysis.** We compare uncertainty calibration with other uncertainty estimation methods, MC-dropout [13] and variational Bayesian neural network (BNN) with mean-field Gaussian weights [34], using the standardized squared error (SSE). For every sample and every dimension  $d = 1, \dots, D$ , we compute  $\text{SSE} = \sum_{i=1, d=1}^{N, D} \{z_i^{(d)}\}^2 / ND$ , where  $z_i^{(d)} = (x_i^{(d)} - \mu^{(d)}) / \sigma^{(d)}$ . If the predictive variance is perfectly calibrated then  $\mathbb{E}(z) = 1$ , thus  $\text{SSE} = 1$ , otherwise  $\text{SSE} > 1$  indicates under-estimated variance,  $\text{SSE} < 1$  indicates over-estimated variance.

In Table 3, the SSE results show that VPT slightly over-estimates its variance, while keeping the best calibration. MC-dropout and BNN underestimate their variances. The hierarchical structure of VPT updates the shrunk leaf probabilities toward their parents when data are sparse. This automatically inflates predictive variance in low-count regions. MC-Dropout [13] and mean-field BNNs [34]

variance often fails to fully propagate to the output, leading to the systematic under-estimation.

Figure 5 visualizes posterior variance estimates from our trained 4-level VPT on MNIST. Higher uncertainty is observed around the central digit region, aligning well with intuition—this area exhibits greater variability across digits. This confirms our model’s capability to provide meaningful uncertainty quantification alongside accurate density estimation.

We emphasize that VPT’s variances are analytic, no Monte-Carlo sampling is required at test time, whereas both baselines need extra computation per input. Thus VPT offers better calibration without extra computational burden.

**Effectiveness of the tree structure.** To evaluate the effectiveness of the hierarchical structure of VPT, we compare it with a learnable histogram (LH) prior. We set the number of bins to  $K = 2^L$  to match the number of nodes in an  $L$ -level VPT. For each dimension  $d$ , we parametrize a increasing sequence  $b_0^{(d)} < b_1^{(d)} < \dots < b_K^{(d)}$ , by setting the lower boundary  $b_0^{(d)}$  and  $b_k^{(d)} = b_{k-1}^{(d)} + \text{softplus}(\delta_{k-1}^d)$ . The parameters  $\delta_{k-1}^d$  are learned jointly with the backbone network, and  $\text{softplus}(\cdot)$  guarantees positive bin widths. Given one dimension  $x^{(d)}$ , we can find the active bin index  $k_*$ , and compute the log-likelihood as  $\log p(x^{(d)}) = \log p_{k_*}^{(d)} - \log(b_{k_*+1}^{(d)} - b_{k_*}^{(d)})$ . All the hyperparameters are kept the same during training as those in VPT.

Table 4 shows the results on UCI datasets with the Block-NAF backbone. The LH prior is used as a plug-in. We observe a clear advantage of VPT over LH, although they share the same number of parameters. We compare LH with VPT on CIFAR-10 with the NICE network. With  $L = 4$ , VPT obtains a result of 2.06 bits-per-dimension, and LH yields 2.57 BPD.

Table 4: Log-likelihoods (standard deviations) on the test set for density estimation using our VPT and the learnable histogram (LH) prior under the same setting ( $L = 4$  and  $K = 2^4$ ) on three real datasets with 5 runs. Higher values indicate better estimation.

Priors	POWER	GAS	BSD300
Learnable histogram (LH)	0.62 <sub>(0.01)</sub>	11.00 <sub>(0.20)</sub>	156.98 <sub>(0.93)</sub>
Variational Pólya tree (VPT)	0.67 <sub>(0.01)</sub>	11.92 <sub>(0.10)</sub>	158.59 <sub>(1.02)</sub>

While a truncated factorized VPT may resemble a learnable histogram in terms of producing a piecewise-constant density, VPTs are more robust. Unlike an LH, where bins are estimated independently, VPT leverages the hierarchical conjugacy, enabling information sharing across scales. Specifically, every Beta-distributed node in the VPT is coupled to its ancestors. This hierarchical shrinkage allows small bins with few data points to borrow statistical strength from parent nodes, automatically adjusting the degree of smoothing. The VPT’s variational posterior inference maintains the joint posterior over tree splits. This enables a more coherent and uncertainty-aware density estimate, especially in regions of sparse data. In contrast, LHs often suffer from overfitting in low-density regions or collapsing of underpopulated bins.

**Interpretation of VPTs.** Besides density estimation, VPT naturally provides probability mass at multiple scales, providing ‘coarse-to-fine’ insights. To better understand what the model has learned, we perform interpolation by linearly mixing the latent representations of MNIST images. Figure 4 illustrates the interpolation paths between pairs of digits under different prior distributions.

Compared to Gaussian and logistic priors, VPT yields meaningful traversals in the latent space. For example, when interpolating from ‘0’ to ‘3’, the intermediate representations pass through recognizable ‘1’. Similarly, interpolating from ‘9’ to ‘8’ reveals a transition through ‘3’. These patterns suggest that, under the VPT prior, the latent space is hierarchically organized, a node

Table 3: Predictive variance calibration with SSE closer to 1 the better.

Datasets	VPT	MC-Dropout	BNN
POWER	0.92	1.30	1.17
GAS	0.90	1.35	1.14
MNIST	0.92	1.27	1.08

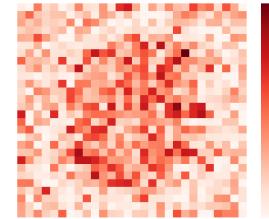


Figure 5: Posterior variance of the trained 4-level VPT model on the MNIST dataset. On each dimension/pixel ( $d = 1, \dots, D$ , where  $D = 28 \times 28$ ), we visualize the mean over the variances of terminal node Beta distribution.

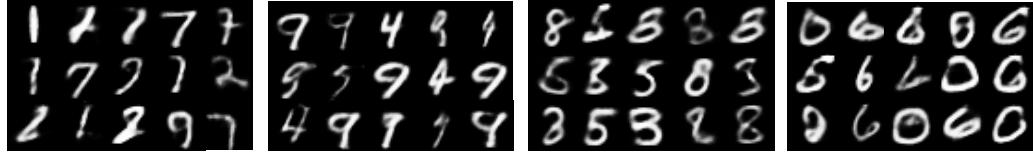


Figure 6: From left to right: images sampled from leaf nodes (nodes 3, 4, 5, 6) of a 3-level VAE-VPT.

corresponding to the digit ‘1’ likely resides between nodes representing ‘0’ and ‘3’, while a node associated with ‘3’ is positioned between those representing ‘9’ and ‘8’. This hierarchical structure offers an interpretable organization of latent representations compared to standard unimodal priors.

## 5 VAEs with the VPT Prior

Beyond the above experiments, we demonstrate the flexibility of our VPT prior by integrating it into a VAE. We use a vanilla VAE architecture with fully connected encoder and decoder networks, each consisting of two hidden layers of 400 neurons and a latent space dimension of 2. By simply replacing the Gaussian prior with our VPT prior, we define a modified ELBO objective as

$$\mathcal{L}_{\text{VPT-VAE}} = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - \lambda \sum_{v \in \text{nodes}} \text{KL}(q_\phi(z_v|x) \| p(z_v)),$$

where the first term is the reconstruction error as in the standard VAE and each node  $v$  has a KL term comparing the approximated posterior  $q_\phi(z_v|x)$  to the VPT prior  $p(z_v)$ .

Specifically, we implement a 3-level VPT prior using a tree with 7 nodes, where node 0 is the root, nodes 1–2 are intermediate nodes, and nodes 3–6 are leaf nodes. After training, we generate images by directly sampling latent variables conditioned on each leaf node (see Figure 6). The leaf nodes naturally cluster visually similar digit images. For instance, node 3 generates digit images resembling ‘1’, ‘2’ and ‘7’, while node 4 produces ‘9’ and ‘4’. Leaf node 5 tends to generate curved digits like ‘3’, ‘5’ and ‘8’, and node 6 generates looped digits such as ‘0’ and ‘6’. This demonstrates that the hierarchical partitioning induced by the VPT prior effectively captures meaningful groupings within the latent space, which enhances the interpretability compared to a VAE with a Gaussian prior.

## 6 Conclusion

We introduce the variational Pólya tree, the first deep generative framework to integrate continuous Pólya tree priors with neural architectures. VPT effectively captures complex, high-dimensional densities and scales to large datasets. While classical BNP models such as Mondrian forests [31] adaptively grow deeper with more data to guarantee universal consistency, we compensate for the fixed tree level  $L$  using expressive neural transformations that allow shallow trees to capture complex structures in latent space. Although a fixed  $L$  may limit asymptotic consistency, our approach maintains key theoretical properties. As supported by Orbanz [37], a truncated PT can still achieve local  $\alpha$ -Hölder continuity at non-dyadic points, preserving approximation capability without requiring strong continuity assumptions. Furthermore, we learn the Beta parameters via variational inference rather than fixing them the priori, enhancing adaptivity while preventing overfitting.

Extensive experiments show that VPT significantly improves density estimation performance across various datasets. Beyond enhanced predictive accuracy, our approach provides meaningful uncertainty quantification and improved interpretability. Our findings demonstrate that VPT delivers competitive performance at different levels of the tree. This research opens promising avenues for advancing density estimation models using Bayesian nonparametric methods.

## Acknowledgements

This work was supported in part by the Research Grants Council of Hong Kong (27206123, 17200125, C5055-24G, and T45-401/22-N), the Hong Kong Innovation and Technology Fund (GHP/318/22GD), the National Natural Science Foundation of China (No. 62201483), and the Guangdong Natural Science Fund (No. 2024A1515011875).

## References

- [1] BART: Bayesian additive regression trees. *The Annals of Applied Statistics*, 4(1):266–298, 2010.
- [2] Naoki Awaya and Li Ma. Hidden markov Pólya trees for high-dimensional distributions. *Journal of the American Statistical Association*, 119(545):189–201, 2024.
- [3] Siavash A. Bigdeli, Geng Lin, L. Andrea Dunbar, Tiziano Portenier, and Matthias Zwicker. Learning generative models using denoising density estimators. *IEEE Transactions on Neural Networks and Learning Systems*, 35(12):17730–17741, 2024.
- [4] Ismaël Castillo. Pólya tree posterior distributions on densities. *Annales de L’Institut Henri Poincaré Section (B) Probability and Statistics*, 53(4):2074–2102, 2017.
- [5] Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, 2016.
- [6] Nicola De Cao, Wilker Aziz, and Ivan Titov. Block neural autoregressive flow. In *Uncertainty in artificial intelligence*, pages 1263–1273, 2020.
- [7] Barry De Ville. Decision trees. *Wiley Interdisciplinary Reviews: Computational Statistics*, 5(6):448–455, 2013.
- [8] Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: Non-linear independent components estimation. *Workshop of International Conference on Learning Representations*, 2014.
- [9] Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real NVP. In *International Conference on Learning Representations*, 2017.
- [10] Finale Doshi, Kurt Miller, Jurgen Van Gael, and Yee Whye Teh. Variational inference for the indian buffet process. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics*, volume 5, pages 137–144, 2009.
- [11] Michael Dusenberry, Ghassen Jerfel, Yeming Wen, Yian Ma, Jasper Snoek, Katherine Heller, Balaji Lakshminarayanan, and Dustin Tran. Efficient and scalable Bayesian neural nets with rank-1 factors. In *International Conference on Machine Learning*, pages 2782–2792, 2020.
- [12] Chris Finlay, Joern-Henrik Jacobsen, Levon Nurbekyan, and Adam Oberman. How to train your neural ODE: the world of Jacobian and kinetic regularization. In *The 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 3154–3164, 2020.
- [13] Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian approximation: representing model uncertainty in deep learning. In *International Conference on Machine Learning*, pages 1050–1059, 2016.
- [14] Philip N Garner and Sibo Tong. A Bayesian approach to recurrence in neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(8):2527–2537, 2020.
- [15] Will Grathwohl, Ricky T. Q. Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. FFJORD: Free-form continuous dynamics for scalable reversible generative models, 2018.
- [16] Manuel Haußmann, Fred A Hamprecht, and Melih Kandemir. Variational Bayesian multiple instance learning with Gaussian processes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6570–6579, 2017.
- [17] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pages 6840–6851, 2020.
- [18] Chin-Wei Huang, David Krueger, Alexandre Lacoste, and Aaron Courville. Neural autoregressive flows. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 2078–2087, 2018.

[19] Kirthevasan Kandasamy, Willie Neiswanger, Jeff Schneider, Barnabas Poczos, and Eric P Xing. Neural architecture search with Bayesian optimisation and optimal transport. *Advances in Neural Information Processing Systems*, 31, 2018.

[20] Diederik P. Kingma and Max Welling. Auto-encoding variational Bayes. In *2nd International Conference on Learning Representations*, 2014.

[21] Diederik P Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. On density estimation with diffusion models. In *Advances in Neural Information Processing Systems*, 2021.

[22] Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

[23] Durk 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, 2016.

[24] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[25] Michael Lavine. Some aspects of Pólya tree distributions for statistical modelling. *The Annals of Statistics*, pages 1222–1235, 1992.

[26] Naiqi Li, Wenjie Li, Yong Jiang, and Shu-Tao Xia. Deep Dirichlet process mixture models. In *Uncertainty in Artificial Intelligence*, pages 1138–1147, 2022.

[27] Zhihang Li, Teng Xi, Jiankang Deng, Gang Zhang, Shengzhao Wen, and Ran He. GP-NAS: Gaussian process based neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11933–11942, 2020.

[28] 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.

[29] R Daniel Mauldin, William D Sudderth, and Stanley C Williams. Pólya trees and random distributions. *The Annals of Statistics*, pages 1203–1221, 1992.

[30] Shaun McDonald and David Campbell. A review of uncertainty quantification for density estimation. *Statistics Surveys*, 15:1 – 71, 2021.

[31] Jaouad Mourtada, Stéphane Gaïffas, and Erwan Scornet. AMF: Aggregated Mondrian forests for online learning. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 83(3):505–533, 2021.

[32] Peter Müller, Fernando Andrés Quintana, Alejandro Jara, and Tim Hanson. *Bayesian Nonparametric Data Analysis*, volume 1. Springer, 2015.

[33] Eric Nalisnick and Padhraic Smyth. Stick-breaking variational autoencoders. In *International Conference on Learning Representations*, 2017.

[34] Radford M Neal. *Bayesian Learning For Neural Networks*, volume 118. Springer Science & Business Media, 2012.

[35] Luis E Nieto-Barajas and Peter Müller. Rubbery Pólya tree. *Scandinavian Journal of Statistics*, 39(1):166–184, 2012.

[36] Junier Oliva, Avinava Dubey, Manzil Zaheer, Barnabas Poczos, Ruslan Salakhutdinov, Eric Xing, and Jeff Schneider. Transformation autoregressive networks. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 3898–3907, 2018.

[37] Peter Orbanz. Projective limit random probabilities on polish spaces. *Electronic Journal of Statistics*, 5:1354–1373, 2011.

[38] Susan M Paddock, Fabrizio Ruggeri, Michael Lavine, and Mike West. Randomized Pólya tree models for nonparametric Bayesian inference. *Statistica Sinica*, pages 443–460, 2003.

- [39] George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. In *Advances in Neural Information Processing Systems*, volume 30, 2017.
- [40] Carl Edward Rasmussen and Christopher KI Williams. *Gaussian Processes for Machine Learning*, volume 1. Springer, 2006.
- [41] Subham Sekhar Sahoo, Aaron Gokaslan, Christopher De Sa, and Volodymyr Kuleshov. Diffusion models with learned adaptive noise. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.
- [42] Sebastian Salazar. VaRT: Variational regression trees. *Advances in Neural Information Processing Systems*, 36, 2024.
- [43] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical Bayesian optimization of machine learning algorithms. *Advances in Neural Information Processing Systems*, 25, 2012.
- [44] Yang Song, Conor Durkan, Iain Murray, and Stefano Ermon. Maximum likelihood training of score-based diffusion models. In *Advances in Neural Information Processing Systems*, volume 34, pages 1415–1428, 2021.
- [45] Ba-Hien Tran, Simone Rossi, Dimitrios Milios, and Maurizio Filippone. All you need is a good functional prior for Bayesian deep learning. *Journal of Machine Learning Research*, 23(74):1–56, 2022.
- [46] Benigno Uria, Iain Murray, and Hugo Larochelle. A deep and tractable density estimator. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32, pages 467–475, 2014.
- [47] Chenyang Zhang and Guosheng Yin. Bayesian nonparametric analysis of restricted mean survival time. *Biometrics*, 79(2):1383–1396, 2023.
- [48] Luping Zhao, Timothy E Hanson, and Bradley P Carlin. Mixtures of polya trees for flexible spatial frailty survival modelling. *Biometrika*, 96(2):263–276, 2009.

## NeurIPS Paper Checklist

### 1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: **[Yes]**

Justification: The abstract and introduction clearly state and accurately reflect our novel contributions and scope.

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

### 2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: **[Yes]**

Justification: We include the discussion of the potential limitations of our work in the conclusions section.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

### 3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [\[Yes\]](#)

Justification: All theoretical results are accompanied by clearly stated assumptions and complete proofs in the appendix.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

#### 4. Experimental result reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [\[Yes\]](#)

Justification: We provide detailed descriptions of data processing and bootstrap-based confidence interval computations to enable full reproducibility.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

#### 5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [\[Yes\]](#)

Justification: All code and data are provided in the anonymized repository, with clear instructions for reproduction.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

## 6. Experimental setting/details

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [\[Yes\]](#)

Justification: We specify data splits, hyperparameters, optimizers, hardware details, in the Implementation Details section.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

## 7. Experiment statistical significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [\[Yes\]](#)

Justification: We report the standard deviations over 5 random runs as described in the main text.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer “Yes” if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

## 8. Experiments compute resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [\[Yes\]](#)

Justification: We detail GPU types, memory for all experiments.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

## 9. Code of ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [\[Yes\]](#)

Justification: We confirm that our research adheres fully to the NeurIPS Code of Ethics guidelines.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

## 10. Broader impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [\[Yes\]](#)

Justification: We discuss positive applications in interpretable deep learning models and uncertainty quantification.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.

- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

## 11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: We determine that our work poses no high-risk assets requiring special release safeguards.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

## 12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We credit all reused code and datasets with explicit citations.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, [paperswithcode.com/datasets](https://paperswithcode.com/datasets) has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

### 13. New assets

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: **[Yes]**

Justification: The code of our model is documented with usage details and licensing information.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

### 14. Crowdsourcing and research with human subjects

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: **[NA]**

Justification: Not applicable as our work does not involve human subject research or crowdsourced data.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

### 15. Institutional review board (IRB) approvals or equivalent for research with human subjects

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: **[NA]**

Justification: Not applicable since our paper does not involve human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

#### 16. Declaration of LLM usage

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: Not applicable as we did not use LLMs in the core method development.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (<https://neurips.cc/Conferences/2025/LLM>) for what should or should not be described.

## A Distributions and KL-Divergences

**Beta distribution.** Let  $X \sim \text{Beta}(\alpha, \beta)$  and  $Y \sim \text{Beta}(\gamma, \delta)$  with  $\alpha, \beta, \gamma, \delta > 0$ , and let  $p$  and  $q$  denote their densities. The KL-divergence between the two Beta distributions is

$$\text{KL}(p \parallel q) = \log \frac{B(\gamma, \delta)}{B(\alpha, \beta)} + (\alpha - \gamma)[\psi(\alpha) - \psi(\alpha + \beta)] + (\beta - \delta)[\psi(\beta) - \psi(\alpha + \beta)],$$

where  $B(\cdot, \cdot)$  is the Beta function and  $\psi(\cdot)$  is the digamma function.

For  $X \sim \text{Beta}(\alpha, \beta)$ , the variance is

$$\text{Var}[X] = \frac{\alpha \beta}{(\alpha + \beta)^2 (\alpha + \beta + 1)}.$$

## Multivariate Gaussian Distribution

The density of a multivariate Gaussian distribution is defined as

$$p(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{\frac{p}{2}} |\boldsymbol{\Sigma}|^{\frac{1}{2}}} \exp \left\{ -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \right\},$$

where  $\boldsymbol{\mu} \in \mathbb{R}^p$  is a  $p$ -dimensional mean vector and  $\boldsymbol{\Sigma} \in \mathbb{R}^{p \times p}$  is the covariance matrix. The KL-divergence between two multivariate normal distributions  $\mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$  and  $\mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$  is

$$\text{KL}(\mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) \parallel \mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)) = \frac{1}{2} \left[ \log \frac{|\boldsymbol{\Sigma}_2|}{|\boldsymbol{\Sigma}_1|} - p + \text{tr}\{\boldsymbol{\Sigma}_2^{-1} \boldsymbol{\Sigma}_1\} + (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1)^\top \boldsymbol{\Sigma}_2^{-1} (\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1) \right].$$

## B Experiment Details

The VPT is implemented with *PyTorch*. All experiments are conducted on a single RTX-3090 GPU.

### Density estimation with real data.

The details of the dataset used in this experiment are summarized in Table A1.

In this experiment, we use Block-NAF [6] as the backbone feature learning network for VPT. We train the backbone network using Adam with Polyak averaging and apply an exponentially decaying learning rate schedule, starting at the learning rate of  $10^{-2}$  with a decay rate of  $\lambda = 0.5$  and a patience of 20 epochs for no improvement. In contrast, our VPT is trained with a constant learning rate of 0.1. All models are trained until convergence, with a maximum of 1,000 epochs, stopping if there is no improvement on the validation set for 100 epochs.

Table A1: Summary of five real UCI datasets.

Datasets	Dimensions	No. Samples
POWER	6	2,049,280
GAS	8	1,052,065
HEPMASS	21	525,123
MINIBOONE	43	36,488
BSDS300	63	1,300,000

**Density estimation and image generation.** We use NICE [8] as the backbone network for this experiment, employing a dequantized version of the data. The architecture consists of a stack of four coupling layers, with a diagonal positive scaling for the last stage. Each coupling function follows the same architecture: five hidden layers of 1,000 units for MNIST, and four layers of 2,000 units for SVHN and CIFAR-10.

The NICE models are trained with Adam with learning rate  $10^{-3}$ , momentum 0.9,  $\beta_2 = 0.01$ ,  $\lambda = 1$  and  $\epsilon = 10^{-4}$ . Our VPT models are trained with Adam with learning rate 0.5.

**Computational complexity analysis.** Our complexity analysis shows that the computational cost of VPT scales primarily with the number of nodes per dimension, which is  $O(2^{LD})$  for a tree of level  $L$  and the feature dimension of  $D$ . In our implementation, we mitigate this cost by choosing a small value of  $L$  (e.g., 2–4) and leveraging the independence across dimensions to parallelize computation. The recursive structure does introduce overhead due to tree traversal and node splitting, while these operations can be optimized via parallel processing across dimensions and nodes.

Table A2: Empirical profiling: training time per epoch for different datasets and configurations.

Datasets	No. Samples	$D$	Model	Tree Level ( $L$ )	Train Time/Epoch
GAS	1,052,065	8	Block-NAF	—	1 min 30 sec
			VPT	4	2 min 10 sec
			VPT	6	2 min 20 sec
POWER	2,049,280	6	Block-NAF	—	3 min
			VPT	4	3 min 50 sec
			VPT	6	4 min
MNIST	60,000	784	NICE	—	30 sec
			VPT	2	35 sec
			VPT	4	50 sec
CIFAR10	60,000	1024	NICE	—	50 sec
			VPT	2	1 min
			VPT	4	1 min 20 sec

Empirical profiling indicates that, when implemented on modern hardware, the additional overhead from VPT is modest compared to that of the normalizing flow backbone. Further optimizations—such as batch processing of tree nodes, caching intermediate results, and efficient tree traversal algorithms—can reduce computational burden even more. Future work will include a detailed benchmark over various  $D$  and  $L$  settings to quantify these improvements.

Furthermore, the additional memory footprint introduced by VPT is negligible, as shown in Table A3. The memory overhead remains below 0.05%, demonstrating that memory usage stays essentially constant when using VPT.

Table A3: Peak GPU memory usage (in gigabytes) during training with Adam and a batch size of 128 using Block-NAF as the backbone. Compared to vanilla Block-NAF with a Gaussian prior, VPTs do not introduce any additional memory overhead.

Datasets	Gaussian Prior	VPT ( $L = 4$ )	VPT ( $L = 6$ )
POWER	0.22	0.22	0.22
GAS	0.22	0.22	0.22
HEPMASS	5.00	5.00	5.00
MINIBOONE	4.04	4.04	4.04
BSD300	19.83	19.83	19.84

It is important to note that the dimensionality  $D$  referenced in the complexity analysis  $\mathcal{O}(2^L D)$  typically corresponds to the latent representation rather than raw pixel dimensions. For image datasets, standard dimensionality reduction techniques such as convolutional layers effectively reduce  $D$ , making it practically manageable even for high-dimensional inputs.

## C Additional Experiment Results on Density Estimation

### Density estimation and image generation on the SVHN dataset.

In addition to the experiments in Section 4.3, we also conduct experiments on the SVHN dataset.

Figure A1 presents the density estimation results evaluated on the test set, while randomly generated images are shown in Figure A2. Similar to our findings with MNIST and CIFAR-10, our VPT enhances density estimation on SVHN without sacrificing perceptual quality.

**MNIST interpolation.** Figure A3 presents more results on the MNIST interpolations. Similar to Figure 4, we can observe that the interpolation images with VPT are more clear and meaningful than those with the traditional Gaussian and logistic prior distributions.

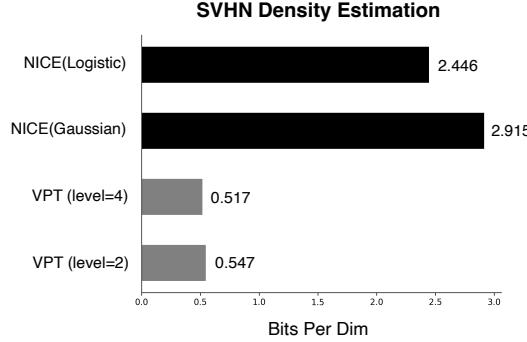


Figure A1: Negative log-likelihood in bits-per-dimension on the SVHN test set.

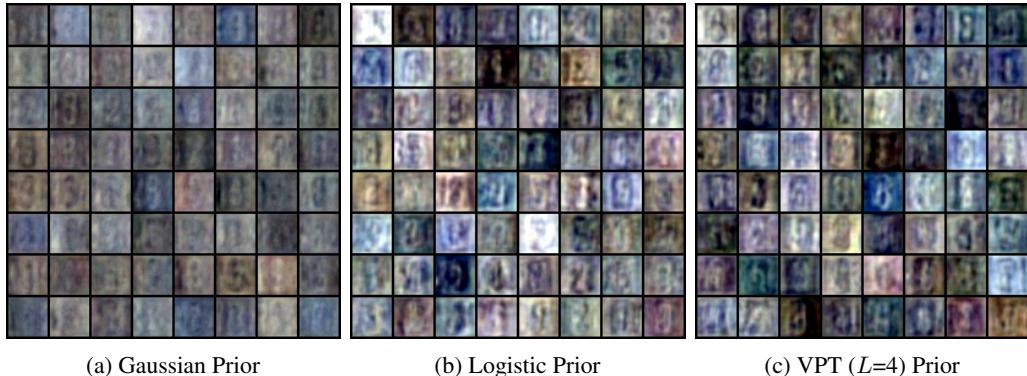


Figure A2: Randomly generated SVHN images with the NICE backbone under different priors.

**Perceptual quality.** Although perceptual quality is not our primary focus, Figure A4 shows that randomly sampling from our VPT prior produces images with clarity comparable to standard Gaussian or logistic priors. This indicates that VPT improves log-likelihood without sacrificing image generation quality, providing an advantageous balance between interpretability, uncertainty quantification, and generation performance.

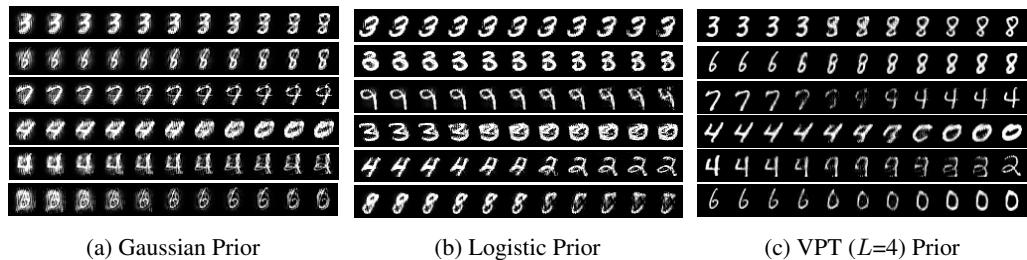
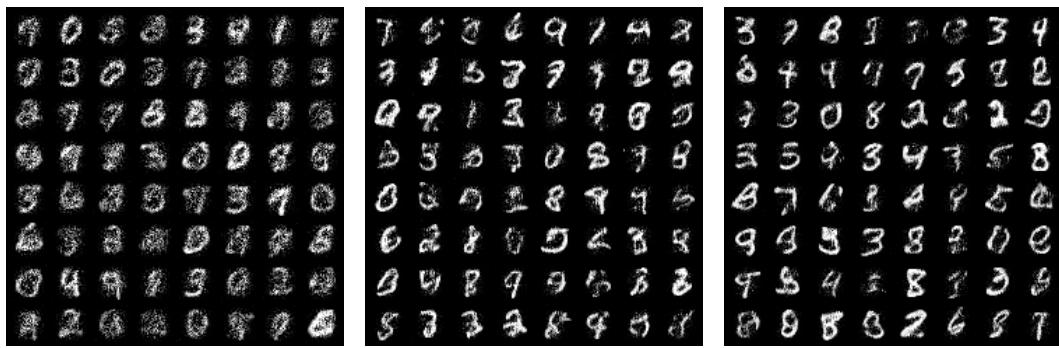


Figure A3: Interpolation on the MNIST images with the NICE backbone under different priors.



(a) Gaussian Prior

(b) Logistic Prior

(c) VPT ( $L=4$ ) Prior

Figure A4: Randomly generated MNIST images with the NICE backbone under different priors.