Approximate Bayesian Inference via Bitstring Representations

Aleksanteri Sladek[1] **Martin Trapp**[1] **Arno Solin**[1]

[1]Department of Computer Science, Aalto University, Espoo, Finland

Abstract

The machine learning community has recently put effort into quantized or low-precision arithmetics to scale large models. This paper proposes performing probabilistic inference in the quantized, discrete parameter space created by these representations, effectively enabling us to learn a continuous distribution using discrete parameters. We consider both 2D densities and quantized neural networks, where we introduce a tractable learning approach using probabilistic circuits. This method offers a scalable solution to manage complex distributions and provides clear insights into model behavior. We validate our approach with various models, demonstrating inference efficiency without sacrificing accuracy. This work advances scalable, interpretable machine learning by utilizing discrete approximations for probabilistic computations.

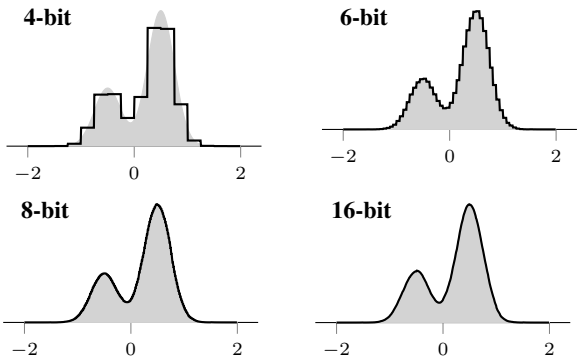


Figure 1: **Capturing a 1D Gaussian mixture** with BitVI with different numbers of bits in the bitstring. Even the 4-bit result serves a practical purpose, while the model saturates around 8 bits when compared to its 16 bit version.

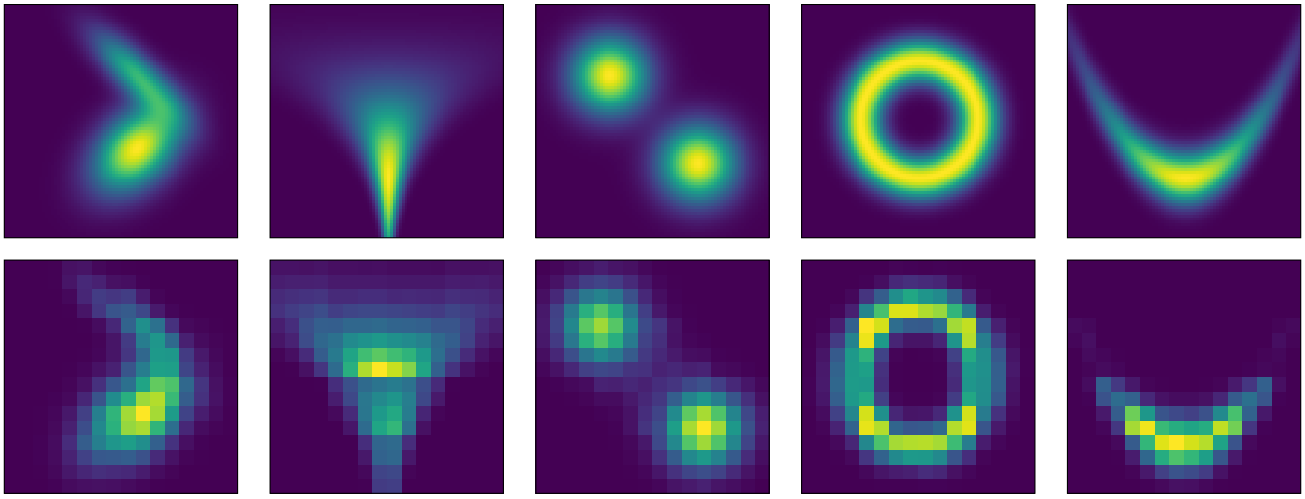
1 INTRODUCTION

Probabilistic inference is central to modern machine learning, providing a principled framework for reasoning under uncertainty. In Bayesian inference, uncertainty is captured through probability distributions over parameters, with Bayes' theorem offering a systematic way to update beliefs with data. However, exact Bayesian inference is often intractable due to the complexity of the integrals involved. Variational inference (VI) [Blei et al., 2017, Jordan et al., 1999, Wainwright and Jordan, 2008] is typically employed as a scalable alternative to Markov chain Monte Carlo (MCMC) methods, enabling inference in high-dimensional models. Despite its success, VI relies on continuous parameterizations and often restrictive Gaussian assumptions, which can introduce representational and computational inefficiencies, particularly in large-scale settings.

To address computational constraints, the machine learning community has increasingly embraced quantization techniques. These methods reduce numerical precision to improve efficiency, leveraging low-bit representations for storage and computation. Many of those can be related to reducing the numerical precision, such as developing tailored low-precision number systems [Gustafson and Yonemoto, 2017, Agrawal et al., 2019] or methods for parameter quantization. Recent works leveraging large-scale mixed-precision FP8 [*e.g.*, Liu et al., 2024], FP4 [Wang et al., 2025], or even 1-bit neural architectures [Ma et al., 2024] have shown innovative low-precision training approaches.

Fig. 1 illustrates how a Gaussian mixture model, typically represented in high-precision floating point, can be equivalently expressed using a low-precision bitstring representation, motivating the feasibility of inference in quantized spaces. These developments suggest that probabilistic inference need not be tied to continuous-valued computations but can instead be formulated in the space of bitstrings.

This work hinges on the fundamental principle that on a computer, continuous values are represented by finite-length bitstrings—that is, a discrete representation. Hence, prob-



(a) Gaussian Mixture (b) Neal's Funnel (c) Two-modal Gaussian (d) Ring (e) Banana

Figure 2: Target densities (upper row) and 4-bit BitVI (lower row) on non-Gaussian 2D density functions. BitVI can model the densities well despite the low bit precision. We include comparisons to full-covariance Gaussian VI in Fig. 11.

ability distributions representable with a computer necessarily possess a representation over finite-length bitstrings. We explore this connection and derive a method to conduct efficient approximate inference through this link.

This work introduces **BitVI**, a novel approach for approximate probabilistic inference in bitstring models. BitVI exploits the inherent discrete nature of number representations to approximate continuous distributions directly in the space of bitstrings. By leveraging probabilistic circuits [Darwiche, 2003, Choi et al., 2020], our method provides a tractable way to learn and perform inference over complex distributions without requiring high-precision representations. Fig. 2 demonstrates how BitVI can model complex distribution with only 4-bit precision.

We validate BitVI across *(i)* standard benchmark densities, demonstrating its ability to approximate known distributions; and *(ii)* Bayesian deep learning in neural network models, where BitVI enables uncertainty quantification. Our results highlight the efficiency and accuracy of BitVI, making it a compelling alternative to traditional inference methods. Our contributions can be summarized as follows.

- **Methodological:** We introduce BitVI, a novel approach for approximate Bayesian inference in bitstring models, leveraging probabilistic circuits for efficient learning and inference.
- **Experimental:** We provide proof-of-concept and benchmarking results on standard test problems as well as Bayesian deep learning tasks, demonstrating the effectiveness of BitVI in practical applications.
- **Insights:** We explore the role of bitstring representations in probabilistic inference and shed light on the trade-offs between model flexibility and quantization.

2 BACKGROUND AND RELATED WORK

The relationship between continuous and discrete representations is fundamental to computational science. At its core, digital computation relies on discrete structures, with real-valued quantities encoded as finite-length bitstrings [Ch. 4 Knuth, 1997]. Floating-point arithmetic provides an approximation to continuous values within this discrete framework, ensuring efficient numerical operations while introducing inherent precision limitations [Ch. 1 Sterbenz, 1974]. In recent years, this foundational connection has gained renewed attention in machine learning, particularly due to advances in quantization and low-precision arithmetic. While these techniques are primarily motivated by hardware constraints, they also present an opportunity: if inference can be formulated directly over discrete bitstring representations, it may unlock new efficiencies in probabilistic modeling.

Bayesian inference provides a principled framework for reasoning under uncertainty, yet exact inference remains intractable in most real-world scenarios. This has led to the development of approximate inference techniques, such as variational inference (VI) [Blei et al., 2017, Jordan et al., 1999, Wainwright and Jordan, 2008]. VI formulates inference as an optimization problem, where a parametric distribution is fitted to approximate the posterior while minimizing the reverse KL divergence. Despite its scalability, VI is often constrained by its reliance on continuous parameterizations, which can introduce numerical instabilities and bias due to restrictive approximations, *e.g.*, mean-field or unimodality assumptions. These limitations are apparent when operating under low-precision, raising the question: *Can we perform inference directly in a discrete representation space?*

Probabilistic circuits (PCs) are a recent framework to study

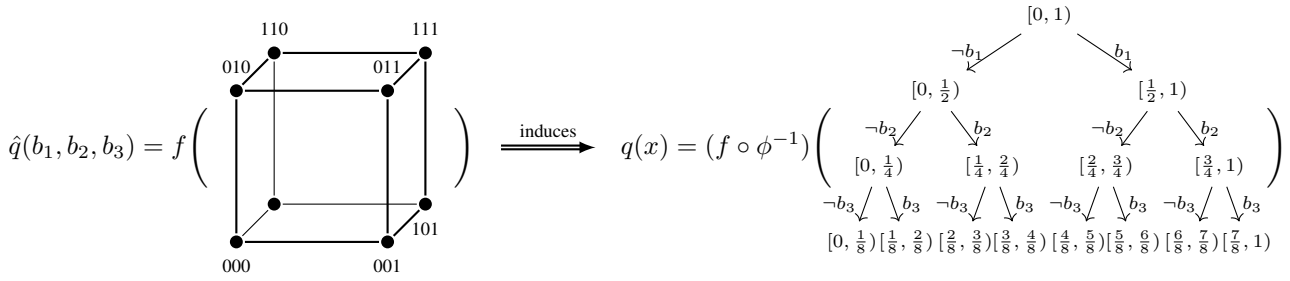


Figure 3: **Illustration of our method:** For the case of fixed-point numbers, we use the bitstring to up to each sum node to index the sum in the circuit. The bitstring can be visualized as a hypercube, and the PC induces a distribution over the fixed-point numbers represented by the bitstring.

tractable representations of complex probability distributions [Choi et al., 2020]. Depending on the structural properties of the PC, certain inference scenarios can be rendered tractable (polynomial in the model complexity) under the circuit while maintaining a high expressivity. While PCs are typically employed for exact probabilistic inference, they have found successful application in approximate Bayesian inference, for example, as surrogate through compilation [Lowd and Domingos, 2010], as variational distribution for structured discrete models [Shih and Ermon, 2020], or in discrete probabilistic programs [Saad et al., 2021]. Our work is closely related to work by Garg et al. [2024], which utilized PCs over bitstring representation for efficient approximate inference in probabilistic programs. This work highlights that PCs are a promising representational framework for approximate Bayesian and uncertainty quantification.

3 METHODS

Given a target density p, we aim to find a variational approximation q that minimizes the divergence of p from q. As commonly done, we will focus on the reverse Kullback–Leibler (KL) divergence of q from p, instead of the forward KL. Moreover, we assume that q takes a parametric form with parameters $\boldsymbol{\theta}$, *i.e.*, $q_{\boldsymbol{\theta}}$. Thus, the goal is to find $\boldsymbol{\theta}$ such that

$$\mathrm{KL}(q_{\boldsymbol{\theta}} \,\|\, p) = \int_{x \in \mathcal{X}} q_{\boldsymbol{\theta}}(x) \log \left(\frac{q_{\boldsymbol{\theta}}(x)}{p(x)} \right) \mathrm{d}x, \qquad (1)$$

is minimized, assuming that $\mathcal{X} \subseteq \mathbb{R}^d$ for some $d \geq 1$.

In general, computing Eq. (1) is intractable for two reasons: *(i)* p is often only known up to an unknown normalization constant Z_p and *(ii)* p and q do not exhibit sufficient structure to render the integration tractable [Wang et al., 2024]. Henceforth, one typically optimizes the evidence lower bound (ELBO), which can be written as

$$\mathcal{L}(q_{\boldsymbol{\theta}}, p) = \mathbb{E}_{x \sim q_{\boldsymbol{\theta}}} \left[\log p(x) \right] + \mathcal{H}\left(q_{\boldsymbol{\theta}} \right), \qquad (2)$$

where $\mathcal{H}(q_{\boldsymbol{\theta}}) = -\mathbb{E}_{x \sim q_{\boldsymbol{\theta}}} \left[\log q_{\boldsymbol{\theta}}(x) \right]$ denotes the entropy of the variational distribution $q_{\boldsymbol{\theta}}$. In case $q_{\boldsymbol{\theta}}$ admits a tractable entropy computation, only the first term in Eq. (2) requires numerical approximation.

When computing either Eq. (1) or Eq. (2) on a computer, each x will inevitably be represented in a discretized form. In fact, every real-valued number is represented by a series of bitstrings and mapped to the real line by a mapping function $\phi \colon \{0,1\}^B \to \mathbb{R}$ given by the chosen number system. Consequently, any distribution p or q represented on a computer can be expressed in terms of a distribution over bitstrings.

Fixed-point representations In this work, we focus on the fixed-point number representation system for bitstrings, which interprets a B-bit bitstring as containing one sign bit, F fractional bits and $B - F - 1$ integer bits. An example of an 8-bit fixed-point bitstring is illustrated in Fig. 4, and its mapping function ϕ is defined in Appendix A.2.

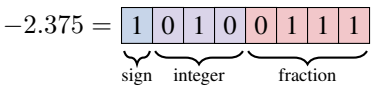


Figure 4: Representation of '-2.375' using an 8-bit fixed-point number system with sign, integer, and fraction bits.

The fixed-point representation was chosen for this work, as its mapping function iteratively partitions an interval (defined by the smallest and largest value ϕ can represent) in $\mathbb{R}$ into fixed-width sub-intervals. As such, this representation is easier to understand and visualize. It is important to note that our approach, however, is not strictly limited to fixed-point representations. For example, the more commonly used floating-point bitstrings can also be used with our method as this representation mainly differs in using variable-width sub-intervals.

In the following, we outline how a continuous distribution can be approximated with a tractable and flexible variational

family defined over its fixed-point bitstring representation.

3.1 BITVI: VARIATIONAL DISTRIBUTIONS OVER BITSTRING REPRESENTATIONS

Let $\hat{q}$ be a distribution over binary strings with probability measure $\hat{Q}$ defined on the measurable space of binary strings $(\mathcal{Y}, \mathcal{A})$ with corresponding σ-algebra $\mathcal{A}$. Further, let $(\mathbb{R}, \mathcal{B})$ be the measurable space of real numbers with Borel σ-algebra $\mathcal{B}$. Define a measurable mapping $\phi\colon \mathcal{Y} \to \mathbb{R}$ that assigns to each binary string a real number according to a specified number system, for example, the fixed point representation. The induced probability measure Q on $(\mathbb{R}, \mathcal{B})$ is the pushforward measure of $\hat{Q}$ through ϕ. Specifically, for any Borel set $B \in \mathcal{B}$ we have $Q(B) = \hat{Q}(\phi^{-1}(B))$ where $\phi^{-1}(B)$ is the pre-image of B under ϕ. Finally, we represent the density q of Q using a (deterministic) probabilistic circuit (PC). This construction is illustrated in Fig. 3 for the case of fixed-point numbers, where we use the bitstring up to each sum node to index the sum in the circuit. For fixed-point representations with infinite precision, this construction is equivalent to probability measures generated by Pólya trees [Ferguson, 1974, Trapp and Solin, 2022].

Definition 3.1 (Deterministic Probabilistic Circuit)**.** *A probabilistic circuit $\mathcal{C}(\boldsymbol{x})$ is a multi-linear function represented by a computational graph consisting of three types of nodes $\mathsf{N}(\boldsymbol{x})$; Sum nodes $\mathsf{S}(\boldsymbol{x}) = \sum_i w_i \mathsf{C}_i(\boldsymbol{x})$ and product nodes $\mathsf{P}(\boldsymbol{x}) = \prod_i \mathsf{C}_i(\boldsymbol{x})$ which compute a function of their child nodes $\mathsf{C}_i(\boldsymbol{x})$ respectively, and leaf nodes consisting of tractable (univariate) functions $\mathsf{L}(x) = p(x)$. The circuit $\mathcal{C}$ characterizes a multivariate probability distribution over random variables $\mathcal{X} = \{X_1, \dots, X_d\}$ by, for example, representing its mass, density, or characteristic function [Yu et al., 2023, Broadrick et al., 2024]. Note that we assume that the circuit is smooth and decomposable [Choi et al., 2020] and refer to Appendix A for details.*

A sum node S is deterministic if for all $\boldsymbol{x}$, only one summand is non-zero. Consequently, $\mathcal{C}$ is deterministic if all sum nodes are deterministic [Choi et al., 2020].

By specifying a $\hat{q}$ over bitstrings and a respective number system, we obtain an induced variational distribution q on the real line. As previously mentioned, our goal is to find a parameterization $\boldsymbol{\theta}$ of our variational distribution such that Eq. (1) is minimal. When representing q using a deterministic PC, the parameters $\boldsymbol{\theta}$ correspond to the collection of weights $\{w_i\}_i$ of the circuit. Note that by construction, the leaf nodes of our circuit model are continuous uniform distributions and, therefore, do not have any additional parameters. The resulting deterministic PC is a tree with depth proportional to the number of bits used in the bitstring representation. Each sum node in the PC represents the decision of a bit and weights correspond to the conditional probability of the respective decision. For example, the probability

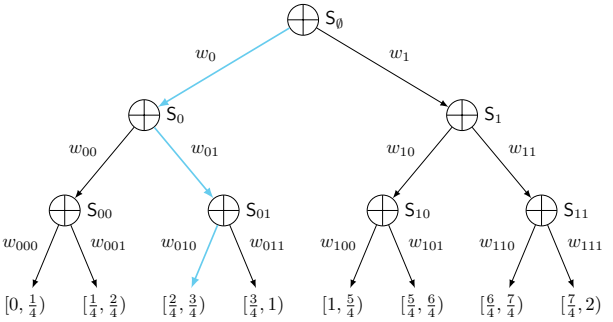


Figure 5: A deterministic PC over a 3-bit fixed-point bitstring. $\mathsf{S}_\emptyset$ encodes the distribution of the full bitstring, and S_ϵ encodes the distribution of the remaining sub-bitstring given ϵ. Intervals $[a, b)$ denote the leaf nodes of the circuit, which are uniform distributions.

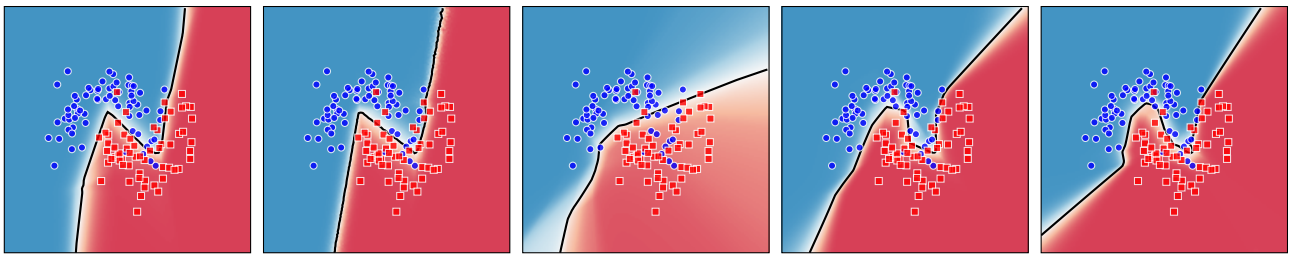
of 0.5 in 3-bit fixed-point number system with one integer bit and no sign-bit, which corresponds to the bitstring 010, is computed by obtaining the bit decisions, *i.e.*, $b_0 = 0$, $b_1 = 1$, and $b_2 = 0$, and evaluating the circuit along the respective path, *i.e.*, $p(x = 0.5) = w_0 w_{01} w_{010} \frac{1}{2^{B_\text{frac}}}$ where $B_\text{frac} = 2$ is the number of fraction bits. Fig. 5 illustrates the decision process represented by the circuit.

Depth Regularization To encourage that q has a smooth density in the limit of infinite precision, we leverage a depth regularization. The depth regularization is based on Pólya tree prior constructions for priors over continuous probability distributions. Specifically, Ferguson [1974] proposed to use a Beta prior on each weight of a Pólya tree with symmetric α-parameter that has a quadratic increase in depth j of the tree, *i.e.*, $\alpha(j) = j^2$. An alternative parameterization is given by Castillo [2017] as $\alpha(j) = 2^j$. Both approaches ensure that the prior probability of uniformly distributed weights increases with depth. We adopt this approach and use Laplace smoothing of the circuit weights with a depth-dependent smoothing factor. In particular, for bit b_j (depth j) with $j \geq 0$ we define each weight for $b_j = 0$ as

$$w_{\epsilon 0} = \frac{v_{\epsilon 0} + c\alpha(j)}{v_{\epsilon 0} + v_{\epsilon 1} + 2c\alpha(j)}, \tag{3}$$

where ϵ denotes a $j-1$ long binary string, $v_{\epsilon 0} > 0$ is an unnormalized weight, and $c > 0$ is a hyperparameter. The weight for $\epsilon 1$ is given analogously.

Computation of the ELBO A particular property of deterministic PCs is that the entropy can be computed in linear time w.r.t. the number of edges of the circuit [Vergari et al., 2021] (see Appendix B for details). As such, we only need to approximate the expected log probability in Eq. (2) using Monte Carlo (MC) integration. To do so, we first use a reparameterization using the inverse CDF transform, which is available analytically in the case of deterministic PCs.



(a) Deterministic (b) MFVI (c) BitVI (2 bits) (d) BitVI (4 bits) (e) BitVI (8 bits)

Figure 6: **Uncertainty quantification in neural networks:** We consider the two moons binary classification problem with an MLP neural network (two hidden layers). The predictive density ( ) shows that BitVI provides both representative uncertainties and good decision boundaries compared to the deterministic and MFVI baselines.

In particular, we reparameterize the ELBO as,

$$\mathcal{L}(q_{\boldsymbol{\theta}}, p) = \mathbb{E}_{u\sim\mathsf{Unif}(0,1)}\left[\log p(F^{-1}_{q_{\boldsymbol{\theta}}}(u))\right] + \mathcal{H}\left(q_{\boldsymbol{\theta}}\right), \quad (4)$$

where $F^{-1}_{q_{\boldsymbol{\theta}}}(\cdot)$ is the inverse CDF transform of $q_{\boldsymbol{\theta}}$. We then generate T samples from a uniform distribution $u^s \sim \mathsf{Unif}(0,1)$ and compute a MC estimate of Eq. (4), *i.e.*,

$$\mathcal{L}(q_{\boldsymbol{\theta}}, p) \approx \frac{1}{T}\sum_{s=1}^{S} \log p(F^{-1}_{q_{\boldsymbol{\theta}}}(u_s)) + \mathcal{H}\left(q_{\boldsymbol{\theta}}\right). \quad (5)$$

Note that Eq. (5) can be computed efficiently.

Remark 3.2. *The inverse CDF transform of $q_{\boldsymbol{\theta}}$ can be computed in linear time w.r.t. the depth of the circuit.* ◁

For a given input y, we can compute the inverse CDF transform of y under $q_{\boldsymbol{\theta}}$ using a series of linear transformations. In particular, sum nodes S compute

$$F^{-1}_{\mathsf{S}_\epsilon}(y) = \begin{cases} F^{-1}_{\mathsf{C}_{\epsilon 1}}\left(\frac{y - w_{\epsilon 0}}{w_{\epsilon 1}}\right) & \text{if } y > w_{\epsilon 0} \\ F^{-1}_{\mathsf{C}_{\epsilon 0}}\left(\frac{y}{w_{\epsilon 0}}\right) & \text{otherwise} \end{cases}, \quad (6)$$

where C denotes a child node of S, *i.e.* a sum or leaf node, and $\epsilon \in \bigcup_{j=0}^{B}\{0,1\}^j$ is a bitstring. If C is a leaf node, we compute the inverse CDF according to the leaf distribution, *i.e.*, $F^{-1}_{\mathsf{L}}(y) = y(b-a)+a$ in case of a continuous uniform distribution $\mathcal{U}(a,b)$. See further details in Appendix B.

Note that the resulting value still requires discretization, and in case of fixed-point numbers needs to be rounded to the nearest fixed-point value. In fact, the bitstring ϵ generated by traversing the circuit in order to compute its inverse CDF already encodes the nearest fixed-point value for y. However, as the discretization operation does not have a well-defined gradient, we resort to the application of the straight-through estimator (STE) [Bengio et al., 2013]. In particular, we compute:

$$x = (\phi(\epsilon) + F^{-1}_{q_{\boldsymbol{\theta}}}(y)) - F^{-1}_{q_{\boldsymbol{\theta}}}(y), \quad (7)$$

where $\phi(\epsilon)$ is the mapping function defined by the number system and the bitstring ϵ is a function of $F^{-1}_{q_{\boldsymbol{\theta}}}$ and indicates the decision taken in Eq. (8).

Representing Multivariate Distributions So far, our induced variational distribution is only defined on the real line (univariate case). To extend the approach to the multivariate case, we considered two approaches: *(i)* a mean-field variational family, and *(ii)* a variational family model with dependencies between dimensions. To represent dependencies between the dimensions, we construct a deterministic PC representing the joint distribution over the bits of all the dimensions. In the case of fixed-point number systems, the resulting circuit model recursively splits the domain into hyper-rectangles by performing axis-aligned splits that alternate between dimensions in the construction. Note that this construction results in a binary tree consisting of 2^{B*D} leaves, where B is the number of bits and D is the number of dimensions. Thus, making it useful in low-dimensional or low-precision settings. However, including conditional independencies in the model can result in substantially more compact representations [Peharz et al., 2020, Garg et al., 2024]. Further details are provided in Appendix A.3.

Applying the inverse CDF reparameterization for multivariate densities modeled with BitVI requires further considerations. In the case of the mean-field approximation, we apply the inverse CDF reparameterization (described above) independently for each dimension. If BitVI represents a variational distribution that models dependencies between dimensions, we employ the inverse of the tree-CDF transformation [Awaya and Ma, 2024], which is a map $\mathbb{R}^D \to [0,1]^D$ where D is the number of dimensions. In particular, for a given input $\boldsymbol{y} \in [0,1]^D$, we compute the inverse tree-CDF transform of $\boldsymbol{y}$ by applying the following axis-aligned linear transformations at each sum node, where $\mathsf{S}_{d,\epsilon_d}$ denotes the sum node for dimension $d \le D$ under bitstring ϵ_d. The axis-aligned transformations are given as:

$$F^{-1}_{\mathsf{S}_{d,\epsilon_d}}(y_d) = \begin{cases} F^{-1}_{\mathsf{C}_1}\left(\frac{y_d - w_{d,\epsilon_d 0}}{w_{d,\epsilon_d 1}}\right) & \text{if } y_d > w_{d,\epsilon_d 0} \\ F^{-1}_{\mathsf{C}_0}\left(\frac{y_d}{w_{d,\epsilon_d 0}}\right) & \text{otherwise} \end{cases}, \quad (8)$$

where with some abuse of notation C_0 denotes the left child of $\mathsf{S}_{d,\epsilon_d}$, which corresponds to a bit value of zero, and C_1 denotes the right child (bit value of one). As we alternate dimensions at each level in the tree, decisions are made only

Table 1: **Bayesian benchmarks:** Negative log predictive density (NLPD$_{\pm \text{std}}$, smaller better) results on the *Bayesian Benchmarks* UCI tasks (5-fold CV). We compare BitVI to Gaussian MFVI and Full-covariance Gaussian VI (FCGVI) on small MLP NN models. The best-performing method for each task is bolded, and multiple methods are bolded based on a paired t-test ($p = 5\%$). We show that BitVI works well on all test cases and is not significantly different from the baselines in most cases, even in the very low-bit range.

Dataset	(n, d)	MFVI	FCGVI	2-BitVI	4-BitVI	8-BitVI
FERTILITY	(100,10)	**0.379**$\pm$0.107	0.406$\pm$0.111	0.728$\pm$0.139	**0.407**$\pm$0.109	**0.406**$\pm$0.142
PITTSBURG-BRIDGES-T-OR-D	(102,8)	**0.345**$\pm$0.168	**0.347**$\pm$0.078	**0.301**$\pm$0.064	**0.352**$\pm$0.082	**0.391**$\pm$0.068
ACUTE-INFLAMMATION	(120,7)	**0.004**$\pm$0.001	0.021$\pm$0.009	**0.006**$\pm$0.002	**0.006**$\pm$0.002	0.684$\pm$0.031
ACUTE-NEPHRITIS	(120,7)	**0.003**$\pm$0.001	0.014$\pm$0.003	**0.002**$\pm$0.000	**0.002**$\pm$0.002	0.051$\pm$0.016
ECHOCARDIOGRAM	(131,11)	**0.446**$\pm$0.167	**0.515**$\pm$0.151	**0.524**$\pm$0.200	**0.435**$\pm$0.095	0.660$\pm$0.132
HEPATITIS	(155,20)	**0.438**$\pm$0.081	**0.447**$\pm$0.116	**0.620**$\pm$0.246	**0.694**$\pm$0.279	**0.427**$\pm$0.085
PARKINSONS	(195,23)	**0.322**$\pm$0.151	**0.284**$\pm$0.109	**0.253**$\pm$0.098	**0.261**$\pm$0.064	**0.289**$\pm$0.061
BREAST-CANCER-WISC-PROG	(198,34)	**0.540**$\pm$0.106	**0.522**$\pm$0.128	0.699$\pm$0.087	**0.584**$\pm$0.073	**0.548**$\pm$0.087
SPECT	(265,23)	**0.614**$\pm$0.067	**0.624**$\pm$0.053	0.801$\pm$0.108	0.807$\pm$0.148	**0.670**$\pm$0.125
STATLOG-HEART	(270,14)	**0.478**$\pm$0.133	**0.488**$\pm$0.156	**0.550**$\pm$0.207	**0.606**$\pm$0.270	**0.478**$\pm$0.147
HABERMAN-SURVIVAL	(306,4)	**0.535**$\pm$0.062	**0.523**$\pm$0.054	**0.531**$\pm$0.042	**0.525**$\pm$0.044	**0.530**$\pm$0.036
IONOSPHERE	(351,34)	**0.288**$\pm$0.094	**0.276**$\pm$0.092	**0.335**$\pm$0.126	**0.459**$\pm$0.217	**0.323**$\pm$0.127
HORSE-COLIC	(368,26)	**0.611**$\pm$0.159	**0.595**$\pm$0.163	0.618$\pm$0.119	0.690$\pm$0.143	**0.576**$\pm$0.103
CONGRESSIONAL-VOTING	(435,17)	**0.670**$\pm$0.093	**0.700**$\pm$0.126	**0.699**$\pm$0.105	**0.704**$\pm$0.108	**0.644**$\pm$0.048
CYLINDER-BANDS	(512,36)	**0.602**$\pm$0.107	**0.633**$\pm$0.050	0.835$\pm$0.222	**0.955**$\pm$0.361	**0.678**$\pm$0.019
BREAST-CANCER-WISC-DIAG	(569,31)	**0.078**$\pm$0.050	**0.108**$\pm$0.029	**0.148**$\pm$0.080	**0.172**$\pm$0.152	**0.155**$\pm$0.097
ILPD-INDIAN-LIVER	(583,10)	**0.547**$\pm$0.059	**0.547**$\pm$0.033	**0.535**$\pm$0.053	**0.518**$\pm$0.032	**0.567**$\pm$0.025
MONKS-2	(601,7)	**0.083**$\pm$0.121	0.607$\pm$0.082	0.563$\pm$0.060	0.656$\pm$0.073	0.666$\pm$0.030
CREDIT-APPROVAL	(690,16)	**0.357**$\pm$0.025	**0.417**$\pm$0.096	0.405$\pm$0.041	**0.358**$\pm$0.026	**0.343**$\pm$0.009
STATLOG-AUSTRALIAN-CREDIT	(690,15)	**0.662**$\pm$0.035	0.650$\pm$0.029	0.764$\pm$0.075	**0.629**$\pm$0.019	**0.626**$\pm$0.019
BREAST-CANCER-WISC	(699,10)	**0.091**$\pm$0.042	**0.105**$\pm$0.041	**0.171**$\pm$0.113	0.168$\pm$0.055	**0.122**$\pm$0.059
BLOOD	(748,5)	**0.483**$\pm$0.058	**0.483**$\pm$0.036	**0.486**$\pm$0.057	**0.478**$\pm$0.043	**0.486**$\pm$0.039
PIMA	(768,9)	0.516$\pm$0.045	**0.507**$\pm$0.042	**0.512**$\pm$0.039	**0.492**$\pm$0.031	**0.492**$\pm$0.042
MAMMOGRAPHIC	(961,6)	**0.428**$\pm$0.039	0.468$\pm$0.044	**0.430**$\pm$0.053	**0.417**$\pm$0.039	**0.423**$\pm$0.049
STATLOG-GERMAN-CREDIT	(1000,25)	**0.547**$\pm$0.066	**0.557**$\pm$0.086	0.651$\pm$0.092	**0.646**$\pm$0.101	0.894$\pm$0.249

based on the 'selected' dimension at each step. Computing the inverse of the tree-CDF transformation can still be performed efficiently, *i.e.*, in $\mathcal{O}(B * D)$ for B bits.

4 EXPERIMENTS

Our experiments are designed to systematically validate the effectiveness of BitVI in performing approximate probabilistic inference over bitstring representations. In Section 4.1, we begin with 2D density estimation to demonstrate the expressiveness of our method in capturing complex non-Gaussian distributions. In Section 4.2, we explore Bayesian deep learning applications by applying BitVI to MLP neural networks (NNs), showcasing its ability to perform effective uncertainty quantification in predictive modeling. We then conduct a series of ablation studies in Section 4.3 to assess the trade-offs between numerical precision and model expressivity, investigating the effect of bitstring depth on performance and the role of hierarchical structure in NNs.

Implementation The method was implemented in Python using the PyTorch library in order to facilitate automatic differentiation, convenient construction of neural network architectures, and fast parallelized training on GPUs. The training was conducted on a high-performance computing cluster with NVIDIA [H,A,V,P]100, K80, and H200 GPUs. As a ballpark, the model training run time for single models in the experiments is measured in the range of minutes for the size of models we consider in these experiments.

4.1 2D DENSITIES

First, we demonstrate the flexibility of our proposed approach in 2D non-Gaussian target distributions. In Fig. 2, we include typical benchmark target densities (mixture, Neal's funnel, two-modal Gaussian, ring, and banana) that we approximate with 4-bit BitVI . Moreover, Fig. 7 shows a comparison for two densities, indicating that BitVI captures the overall density and cross-dependencies well, with approximation quality increasing with the number of bits. Fig. 11 in the Appendix shows comparisons to the remaining densities.

4.2 MLP NEURAL NETWORK MODELS

We experiment with probabilistic inference in multi-layer perceptron (MLP) neural network (NN) models. For simplicity, we use similar neural network architectures in all the NN experiments. We use two hidden layers in all experiments, only varying the number of units. Additionally, we use the layer norm to ensure weight scaling.

Fig. 6 shows an uncertainty quantification example. We consider the two moons binary classification problem with an MLP neural network ([8,8] hidden units). The predictive density shows that BitVI provides both representative uncertainties and good decision boundaries compared to the deterministic and mean-field Gaussian VI baselines.

To give a more quantitative treatment to MLP NN modeling tasks, we use the *Bayesian Benchmarks*[1] community

[1] `github.com/secondmind-labs/bayesian_benchmarks`; originally by Salimbeni *et al.*

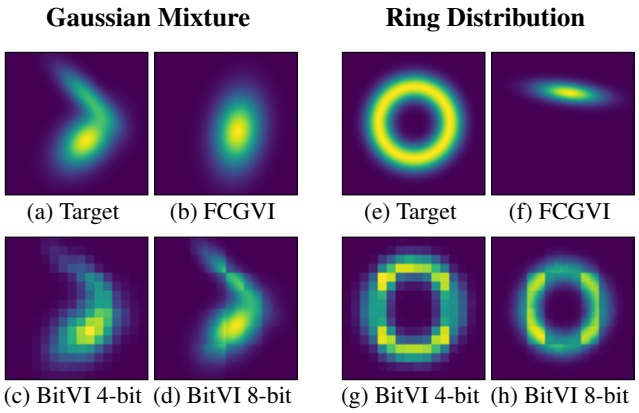


Figure 7: Comparison of 4-bit/8-bit BitVI against full-covariance Gaussian VI (FCGVI) on 2D non-Gaussian target distributions. A full comparison on all target distributions is given in Fig. 11. BitVI captures the overall density and cross-dependencies better than FCGVI.

suite meant for benchmarking Bayesian methods in machine learning. Bayesian benchmarks include common evaluation data sets (typically from UCI [Kelly et al., 2025]) and make it possible to run a large number of comparisons under a fixed evaluation setup. We evaluate our approach in binary classification, and for an interesting probabilistic treatment, we include small-data binary classification tasks with $100 \leq n \leq 1000$ data samples (25 data sets). We follow the standard setup of input point normalization and splits in the evaluation suite. Additional details on the NN architectures and evaluation setup can be found in Appendix C.2.

Table 1 shows the results for BitVI (with 2, 4, and 8 bits), mean-field Gaussian VI (MFVI), and full-covariance Gaussian VI (FCGVI). Our approach consistently performs competitively with the standard variational inference baselines, even in the low-bit regime. Notably, in most data sets, BitVI with 4-bit and 8-bit representations achieves comparable performance to MFVI and FCGVI, demonstrating that probabilistic inference can be effectively conducted over bitstring representations without significant loss in predictive power. Even at 2-bit precision, BitVI remains viable in several cases. Yet, the results also suggest that more flexible probabilistic modeling in this neural network setting might not be needed, as the 8-bit models show very little or any benefits over the 4-bit models.

4.3 ABLATION STUDIES

Increasing Complexity of Target Distribution We consider an ablation study where we control the target distribution complexity. For this, we constructed a mixture of equidistant Gaussians and assessed the entropy of BitVI under varying numbers of bits under three different amounts of variance for each Gaussian. Fig. 9 shows the fitted results of BitVI (black) with 16 bits for target distributions with increasing complexity (gray) alongside the entropy of BitVI under varying number of bits. The entropy (lower figures) shows the cut-off for number of bits needed to represent each target, indicating that BitVI naturally exhibits a parsimonious behaviour.

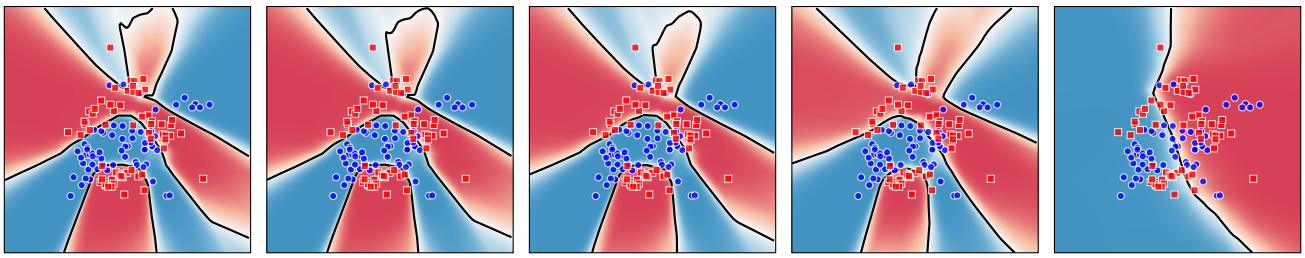
Table 2: The trade-off between NN model complexity (units in hidden layers) and bitstring length (2–12 bits). The negative log predictive density (NLPD, smaller better) on the two moons data suggests that even low bit depth models perform well, and the dominating factor in expressivity is the number of units in the NN. See Appendix D for ACC/ECE.

Bitstring depth ↓ / Increasing NN complexity →	[4, 4]	[6, 6]	[8, 8]	[10, 10]	[12, 12]	[14, 14]	[16, 16]
2	0.36	0.35	0.35	0.32	0.33	0.3	0.29
3	0.37	0.36	0.26	0.34	0.27	0.24	0.25
4	0.38	0.32	0.31	0.3	0.27	0.28	0.24
5	0.35	0.32	0.36	0.29	0.27	0.25	0.25
6	0.34	0.34	0.37	0.3	0.28	0.25	0.24
7	0.31	0.3	0.3	0.26	0.28	0.25	0.24
8	0.33	0.31	0.25	0.3	0.29	0.26	0.26
9	0.36	0.32	0.32	0.33	0.26	0.23	0.25
10	0.33	0.35	0.3	0.3	0.25	0.26	0.24
12	0.37	0.29	0.35	0.35	0.26	0.27	0.24

Trade-off Between Model Complexity and Bitstring Depth For NN applications, an interesting question is whether fine-grained numerical accuracy is needed to represent the model weights in the first place. Recent advances in large-scale model training and inference suggest that rather than numerical accuracy, the models benefit from more parameters, which enable further flexibility. Hence, we study whether the models benefit from higher numerical granularity w.r.t. probabilistic treatment.

In Table 2, we vary both the NN complexity (units in the two hidden layers) and the bitstring length. We consider 2–12-bit models (with only fractional bits). The negative log predictive density (NLPD, smaller better) on the two moons data suggests that even low bit depth models perform well, and the dominating factor in expressivity is the number of units in the NN. In Appendix D, we include similar tables for both accuracy and expected calibration error (ECE).

Do Bitstrings Capture Hierarchies in NNs? Finally, we use a neural network model to study the hierarchies captured by BitVI. We start from a 10-bit NN BitVI results on the Banana binary classification data set and gradually decrease the fractional precision of the trained model, chopping off more granular levels of the model. Fig. 8 shows the results for 10, 8, 6, 4, and 2-bit models (2 integer bits each, except for the 2-bit model). Even the 4-bit model (2 integer bits and 1 fractional bit) captures the overall structure well, whereas the 2-bit model (with no integer bits; only a sign bit and a fraction bit) struggles.



(a) 10-bit (b) 8-bit (c) 6-bit (d) 4-bit (e) 2-bit

Figure 8: **Chopping the banana:** We start from a 10-bit NN BitVI results on the Banana binary classification data set and gradually decrease the fractional precision of the trained model. The low-bit models up to 4 bits capture the overall structure well. This is further confirmed by the results in Table 2.

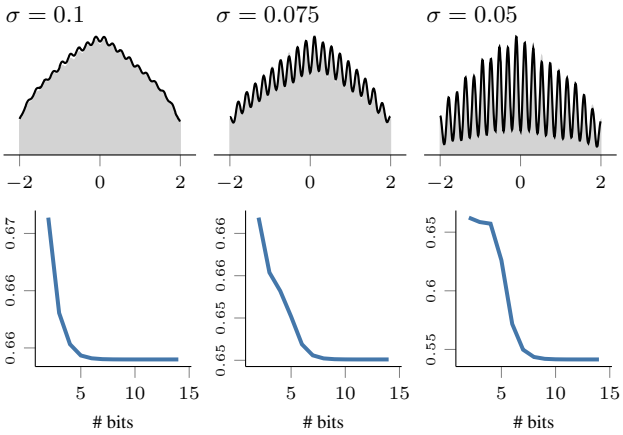


Figure 9: Ablation result of BitVI (black) for target distributions with increasing complexity (gray) and the precision used by the variational distribution to represent the target. The entropy (lower figures) shows the cut-off for bitstring depth needed to represent each target.

5 DISCUSSION AND CONCLUSION

In this work, we introduced **BitVI**, a novel approach for approximate Bayesian inference that operates directly in the space of discrete bitstring representations. By leveraging (deterministic) probabilistic circuits as the representational framework, we demonstrated that inference can be performed directly on bitstring representations of number systems, enabling effective approximate inference and uncertainty quantification. Our approach presents a paradigm shift by learning a rich variational approximation induced by a variational family on bitstring representations without relying on high-precision representations. Our experiments showcased the flexibility of BitVI across different settings: In Section 4.1, we illustrated its ability to approximate complex non-Gaussian densities; and in Section 4.2, we demonstrated its effectiveness in Bayesian deep learning, where it provided robust uncertainty estimates while maintaining computational efficiency.

Beyond demonstrating feasibility, these results highlight that flexible approximate Bayesian inference does not need to be constrained to continuous-valued computations but can be reformulated in a fully discrete manner. Moreover, our results further highlight the potential of using probabilistic circuits as the representational framework for approximate inference. A key strength derived from using probabilistic circuits is the tractability of BitVI which we exploited for computing the entropy of our model in closed-form. While BitVI provides a promising direction for flexible variational inference, several limitations remain.

Limitations To scale to high-dimensional settings, our approach employs a mean-field approximation to the posterior. This limitation arises from our tree construction, which considers dependencies between all bits and dependencies between all dimensions if no mean-field assumption is made. In practical applications, modeling all dependencies is likely unnecessary and introduces an excessive computational and memory burden. Therefore, a promising future direction is to leverage more compact representations such [Peharz et al., 2020]. For the same reason, our approach also currently introduces many parameters to be optimized, which can result in further challenges for high-dimensional settings. Lastly, our experiments focused only on fixed-point representations. Exploiting the representational power of floating-point representations is a promising future avenue.

The codes and resources for BitVI are available on GitHub: `github.com/AaltoML/bitvi`.

Acknowledgements

A. Solin acknowledges funding from the Research Council of Finland (grant number 339730). M. Trapp acknowledges funding from the Research Council of Finland (grant number 347279). A. Sladek acknowledges funding from the Finnish Doctoral Program Network in Artificial Intelligence (AI-DOC, decision number VN/3137/2024-OKM-6). We acknowledge the computational resources provided by the Aalto Science-IT project. We thank the reviewers and the area chair for their constructive feedback.

References

Ankur Agrawal, Bruce M. Fleischer, Silvia M. Mueller, Xiao Sun, Naigang Wang, Jungwook Choi, and Kailash Gopalakrishnan. Dlfloat: A 16-b floating point format designed for deep learning training and inference. In *ARITH*, pages 92–95. IEEE, 2019.

Naoki Awaya and Li Ma. Unsupervised tree boosting for learning probability distributions. *Journal of Machine Learning Research*, 25(198):1–52, 2024.

Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, abs/1607.06450, 2016.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017.

Oliver Broadrick, Honghua Zhang, and Guy Van den Broeck. Polynomial semantics of tractable probabilistic circuits. In *40th Conference on Uncertainty in Artificial Intelligence (UAI)*, Proceedings of Machine Learning Research, pages 418–429. PMLR, 2024.

Ismaël Castillo. Pólya tree posterior distributions on densities. *Annales de l'Institut Henri Poincaré, Probabilités et Statistiques*, 53(4):2074 – 2102, 2017.

YooJung Choi, Antonio Vergari, and Guy Van den Broeck. Probabilistic circuits: A unifying framework for tractable probabilistic models. Technical report, University of California, Los Angeles (UCLA), 2020.

Adnan Darwiche. A differential approach to inference in bayesian networks. *Journal of the ACM (JACM)*, 50(3): 280–305, 2003.

Thomas S. Ferguson. Prior Distributions on Spaces of Probability Measures. *The Annals of Statistics*, 2(4):615 – 629, 1974.

Poorva Garg, Steven Holtzen, Guy Van den Broeck, and Todd Millstein. Bit blasting probabilistic programs. *Proceedings of the ACM on Programming Languages (PACMPL)*, 8, 2024.

John L. Gustafson and Isaac T. Yonemoto. Beating floating point at its own game: Posit arithmetic. *Supercomputing Frontiers and Innovations*, 4(2):71–86, 2017.

Michael I. Jordan, Zoubin Ghahramani, Tommi S. Jaakkola, and Lawrence K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37: 183–233, 1999.

Markelle Kelly, Rachel Longjohn, and Kolby Nottingham. The UCI machine learning repository. `https://archive.ics.uci.edu`, 2025.

Donald E. Knuth. *The Art of Computer Programming: Seminumerical Algorithms*, volume 2. Addison-Wesley Professional, 1997.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Daniel Lowd and Pedro Domingos. Approximate inference by compilation to arithmetic circuits. In *Advances in Neural Information Processing Systems 23 (NeurIPS)*, pages 1477–1485. Curran Associates, 2010.

Shuming Ma, Hongyu Wang, Lingxiao Ma, Lei Wang, Wenhui Wang, Shaohan Huang, Li Dong, Ruiping Wang, Jilong Xue, and Furu Wei. The era of 1-bit LLMs: All large language models are in 1.58 bits. *arXiv preprint arXiv:2402.17764*, 2024.

Robert Peharz, Steven Lang, Antonio Vergari, Karl Stelzner, Alejandro Molina, Martin Trapp, Guy Van den Broeck, Kristian Kersting, and Zoubin Ghahramani. Einsum networks: Fast and scalable learning of tractable probabilistic circuits. In *37th International Conference on Machine Learning (ICML)*, pages 7563–7574. PMLR, 2020.

Feras A. Saad, Martin C. Rinard, and Vikash K. Mansinghka. SPPL: probabilistic programming with fast exact symbolic inference. In *42nd International Conference on Programming Language Design and Implementation (ACM/SIGPLAN)*, pages 804–819, 2021.

Andy Shih and Stefano Ermon. Probabilistic circuits for variational inference in discrete graphical models. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*. Curran Associates, 2020.

Pat H. Sterbenz. *Floating-point Computation*. Prentice-Hall Series in Automatic Computation, 1974.

Martin Trapp and Arno Solin. On priors in Bayesian probabilistic circuits and multivariate pólya trees. In *The 5th Workshop on Tractable Probabilistic Modeling*, 2022.

Martin Trapp, Robert Peharz, Hong Ge, Franz Pernkopf, and Zoubin Ghahramani. Bayesian learning of sum-product networks. In *Advances in Neural Information Processing Systems 32 (NeurIPS)*, pages 6344–6355. Curran Associates, 2019.

Antonio Vergari, YooJung Choi, Anji Liu, Stefano Teso, and Guy Van den Broeck. A compositional atlas of tractable circuit operations for probabilistic inference. In *Advances in Neural Information Processing Systems 34 (NeurIPS)*, pages 13189–13201. Curran Associates, 2021.

Martin J. Wainwright and Michael I. Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2): 1–305, 2008.

Benjie Wang, Denis Deratani Mauá, Guy Van den Broeck, and YooJung Choi. A compositional atlas for algebraic circuits. In *Advances in Neural Information Processing Systems 38 (NeurIPS)*. Curran Associates, 2024.

Ruizhe Wang, Yeyun Gong, Xiao Liu, Guoshuai Zhao, Ziyue Yang, Baining Guo, Zhengjun Zha, and Peng Cheng. Optimizing large language model training using FP4 quantization. *arXiv preprint arXiv:2501.17116*, 2025.

Zhongjie Yu, Martin Trapp, and Kristian Kersting. Characteristic circuits. In *Advances in Neural Information Processing Systems 36 (NeurIPS)*. Curran Associates, 2023.

Approximate Bayesian Inference via Bitstring Representations (Supplementary Material)

Aleksanteri Sladek[1] **Martin Trapp**[1] **Arno Solin**[1]

[1]Department of Computer Science, Aalto University, Espoo, Finland

A TECHNICAL DETAILS

A.1 PROBABILISTIC CIRCUITS

We will briefly review the main concepts related to probabilistic circuits (PC), relevant for this work.

Definition A.1 (scope of a node)**.** *The scope of a node* N *is the set of variables it depends on. The scope for a given node* N *is denoted as* $\psi(\mathsf{N})$*. See [Trapp et al., 2019] for details.*

Definition A.2 (support of a node)**.** *The support of a node* $\mathsf{N}(\boldsymbol{x})$ *is notated as* $\mathrm{supp}(\mathsf{N})$*, and is defined as* $\mathrm{supp}(\mathsf{N}) = \{\boldsymbol{x} \in \mathcal{X} \mid \mathsf{N}(\boldsymbol{x}) > 0\}$*. It is the set of values in* $\mathcal{X}$ *for which the node computes a non-zero value.*

Definition A.3 (smooth & decomposable circuit)**.** *A sum node is smooth if its children have the same scope. A product node is decomposable if its children have pairwise disjoint scopes. A circuit is smooth (resp. decomposable) if all its sum nodes are smooth (resp. product nodes are decomposable).*

In this work, we only consider circuits that fullfill both smoothness and decomposability conditions as they both are required to render common inference tasks, such as density evaluation and marginalisation, tractable.

A.2 FIXED-POINT REPRESENTATION BITSTRINGS

In this work we focus on fixed-point bitstrings. A B-bit bitstring $\boldsymbol{b} = \{b_{(B-1)}, b_{(B-2)}, \ldots, b_{B_0}\}$ is defined as having 1 sign bit, F fractional bits and $B - F - 1$ integer bits. The mapping function ϕ is defined as

$$\phi(\boldsymbol{b}) = (-1)^{b_k} \left(\sum_{i=0}^{k-1} b_i 2^{i-F} \right). \tag{9}$$

Fixed-point bitstrings encode fractional values as negative powers of two, and integer values as positive powers of two.

A.3 MULTIVARIATE BITSTRING REPRESENTATIONS

As outlined in the main text, for multivariate distributions, we generate a circuit model that represents a distribution over hyper-rectangles. Let Ω denote the domain of the distribution, we recursively construct a dyadic partition of the domain into measurable subsets. This process is done by selecting a splitting dimension at each level of the tree and splitting the hyper-rectangle according to the number system representation, *i.e.*, in the middle for fixed-point numbers. At the next level, we select a splitting dimension our of the remaining dimension (those that have not been split yet) and split the hyper-rectange accordingly. We make sure each dimension has been split in the process, before restarting the splitting. The construction ends if each dimension has been split B many times, where B is the number of bits used in the number system. Fig. 10 illustrates the recursive splitting of the input domain Ω into sub-domains (hyper-rectangles).

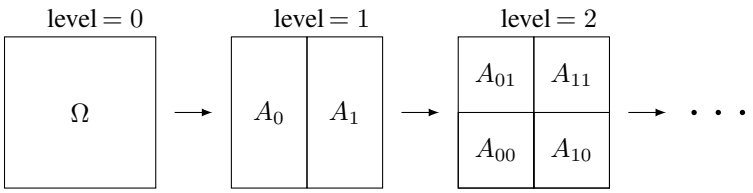


Figure 10: Illustration of the iterative axis-aligned splitting of the domain into hyper-rectangles (sub-domains) by the circuit.

B DERIVATIONS

B.1 BITVI DETERMINISTIC PROBABILISTIC CIRCUIT CONSTRUCTION

In this section, we provide a comprehensive account of our deterministic probabilistic circuit $\mathcal{C}$ construction that defines a probability density function over the bitstring representation of a single continuous valued random variable X. Without loss of generality, we assume $X \in (0, 1]$. Let $\mathcal{C}(x)$ represent a probabilistic circuit such that $\mathcal{C} = (\mathcal{G}, \boldsymbol{\theta})$ where $\mathcal{G}$ is a computational graph parameterized by parameters in $\boldsymbol{\theta}$. Hence, $X \sim \mathcal{C}$.

Let $\mathcal{G}$ be a binary tree structure $\mathcal{T}$. The set of non-leaf nodes of $\mathcal{T}$ is denoted as $\mathcal{N}(\mathcal{T})$, and the set of leaf nodes is denoted as $\mathcal{L}(\mathcal{T})$. Furthermore, let every node $\mathsf{N} \in \mathcal{N}(\mathcal{T})$ be a sum node S. Note that since $\mathcal{T}$ is a binary tree structure, this means that every sum node $\mathsf{S} \in \mathcal{N}(\mathcal{T})$ has exactly two children.

Definition B.1 (Binary Partitioning Tree Structure)**.** *In this work we utilize a binary tree structure $\mathcal{T}$ that serves as the computational graph of our model $\mathcal{C}$. A key aspect of this structure is that it represents a finite-depth partitioning of the support of the distribution that $\mathcal{C}$ represents. Let us consider a case where $\mathcal{C}$ has support* $\text{supp}(\mathcal{C}) = A = (a, b]$, *where* $a, b \in \mathbb{R}$, *and* $a < b$. *This entails that this is the support of the root node* $\mathsf{N}(x)$ *of* $\mathcal{T}$. *Next, we define the supports of the child nodes* $ch(\mathsf{N}) = \{\mathsf{N}_0, \mathsf{N}_1\}$ *of* N *to be a* mutually exclusive partitioning *of* A. *Let* $c \in A$ *and* $a \leq c < b$. *In our case, we set* $c = \frac{a+b}{2}$ *(mid-point), but* c *can be any value* $x \in A$. *Now, the supports of the child nodes are defined as* $A_0 = \text{supp}(\mathsf{N}_0) = (a, c]$ *and* $A_1 = \text{supp}(\mathsf{N}_1) = (c, b]$. *Note that* $A_0 \cap A_1 = \emptyset$ *and* $A_0 \cup A_1 = A$. *We also define the sets* $\mathcal{A}_0 = \{A\}$ *and* $\mathcal{A}_1 = \{A^0, A^1\}$. *The same procedure is then recursively applied to each partition* $A_{\boldsymbol{b}} \in \mathcal{A}_1$ *to create a new set of mutually exclusive partitions* $\mathcal{A}_2$. *This procedure continues until the depth of the tree* $\mathcal{T}$. *Note that this method for partitioning the support at the mid-point of each interval arises from using the fixed-point representation system.*

Definition B.2 (Deterministic Sum Nodes)**.** *Let us define each sum node* $\mathsf{S} \in \mathcal{N}(\mathcal{T})$*:*

$$\mathsf{S}_{\boldsymbol{b}}(x) = \begin{cases} w_L^{\boldsymbol{b}} \mathsf{N}_{\boldsymbol{b}0}(x) + w_R^{\boldsymbol{b}} \mathsf{N}_{\boldsymbol{b}1}(x) & \text{if } x \in (a^{\boldsymbol{b}}, b^{\boldsymbol{b}}] \\ 0 & \text{else} \end{cases}, \tag{10}$$

where $(a^{\boldsymbol{b}}, b^{\boldsymbol{b}}] = supp(\mathsf{S}_{\boldsymbol{b}})$. *Note also that* $w_L^{\boldsymbol{b}}$ *and* $w_R^{\boldsymbol{b}}$ *are parameterized weights associated to sum node* $\mathsf{S}_{\boldsymbol{b}}$, *with the properties* $w_R^{\boldsymbol{b}} + w_L^{\boldsymbol{b}} = 1, w_L^{\boldsymbol{b}} > 0, w_R^{\boldsymbol{b}} > 0$. *Note that scalars* $a^{\boldsymbol{b}}$ *and* $b^{\boldsymbol{b}}$ *define the* support *of* $\text{supp}(\mathsf{S}_{\boldsymbol{b}})$ *of node* $\mathsf{S}_{\boldsymbol{b}}$, *meaning* $\mathsf{S}_{\boldsymbol{b}}(x)$ *only has non-zero probability density within this region, as shown in Eq.* (10). *Furthermore* $\int_{a^{\boldsymbol{b}}}^{b^{\boldsymbol{b}}} \mathsf{S}_{\boldsymbol{b}}(x) = 1$ *since* $w_R^{\boldsymbol{b}} + w_L^{\boldsymbol{b}} = 1$ *and* $\int_{x \in \text{supp}(\mathsf{N})} \mathsf{N}(x) = 1\mathrm{d}x$ *for all* $\mathsf{N} \in \mathcal{T}$. *In other words, the circuit is normalized and representes a valid probability distribution at the root.*

A note on bitstring notation: *Here* $\boldsymbol{b}$ *denotes a vector of bits, or a* bitstring, $b_1 b_2 \ldots b_d$ *with commas and bracket notation omitted for notational convenience and convention. This bitstring represents the path to the current node from the root, with bit value* 0 *denoting a left child and bit value* 1 *a right child. bitstring* $\boldsymbol{b} = \{\}$ *denotes the empty bitstring (i.e. node* S*), and is reserved for denoting the root node of* $\mathcal{T}$. $\mathsf{N}_{\boldsymbol{b}i}$ *denotes a node in* $\mathcal{T}$, *with* $i = 0$ *for the left child of* $\mathsf{S}_{\boldsymbol{b}}$ *and* $i = 1$ *denotes the right child of* $\mathsf{S}_{\boldsymbol{b}}$. $\mathsf{N}_{\boldsymbol{b}i}$ *may be either a* S *or a* L *depending of if* $\mathsf{N} \in \mathcal{N}(\mathcal{T})$ *or* $\mathcal{L}(\mathcal{T})$. *Importantly, the child nodes of* $\mathsf{S}_{\boldsymbol{b}}$ *split its support* $a^{\boldsymbol{b}}$ *and* $b^{\boldsymbol{b}}$ *into sub-intervals* $A_L = (a^{\boldsymbol{b}0}, b^{\boldsymbol{b}0}]$ *and* $A_R = (a^{\boldsymbol{b}1}, b^{\boldsymbol{b}1}]$, *where* $A_L \cap A_R = \emptyset$.

Definition B.3 (Uniform Leaf Nodes)**.** *We define every leaf node* $\mathsf{L}_{\boldsymbol{b}}(x) \in \mathcal{L}(\mathcal{T})$ *as a continuous uniform distribution:*

$$\mathsf{L}_{\boldsymbol{b}}(x) = \begin{cases} \frac{1}{b^{\boldsymbol{b}} - a^{\boldsymbol{b}}} & \text{if } x \in (a^{\boldsymbol{b}}, b^{\boldsymbol{b}}] \\ 0 & \text{else} \end{cases}, \tag{11}$$

where $(a^{\boldsymbol{b}}, b^{\boldsymbol{b}}] = supp(\mathsf{L}_{\boldsymbol{b}})$.

B.2 INVERSE-CDF DERIVATION

This section derives the inverse cumulative density function (CDF) of our circuit construction. Recall that the inverse-CDF was presented in Eq. (8) as a pair of local moves that can be iteratively applied at each depth of the circuit, starting from the root and traversing downwards. We begin by showing an example of the PDF and CDF for a simple circuit construction. Then, we show the computation for the inverse of the CDF function, and extend this to a more general set of rules from there.

To start, we show in the following example what the PDF of a simple (shallow) circuit construction looks like. In this case, we let $\mathcal{T}$ be a depth 3 tree structure i.e. a distribution over 3-bits. The PDF function is then

Example B.4 (PDF Example)**.**

$$\begin{aligned}
\mathcal{C}(x) &= \mathsf{S}(x) && (12)\\
&= w_L \mathsf{S}_0(x) + w_R \mathsf{S}_1(x) && (13)\\
&= w_L \left(w_L^0 \mathsf{S}_{00}(x) + w_R^0 \mathsf{S}_{01}(x)\right) + w_R \left(w_L^1 \mathsf{S}_{10}(x) + w_R^1 \mathsf{S}_{11}(x)\right) && (14)\\
&= w_L \left(w_L^0 \left[w_L^{00} \mathsf{L}_{000}(x) + w_R^{01} \mathsf{L}_{001}(x)\right] + w_R^0 \left[w_L^{01} \mathsf{L}_{010}(x) + w_R^{01} \mathsf{L}_{011}(x)\right]\right) && (15)\\
&\quad + w_R \left(w_L^1 \left[w_L^{10} \mathsf{L}_{100}(x) + w_R^{10} \mathsf{L}_{101}(x)\right] + w_R^1 \left[w_L^{11} \mathsf{L}_{110}(x) + w_R^{11} \mathsf{L}_{111}(x)\right]\right) && (16)
\end{aligned}$$

◁

The PDF function can then be integrated up to some variable x to define the CDF of the circuit construction. Let $F_{\mathcal{C}}$ represent the CDF of $\mathcal{C}$. Then,

Example B.5 (CDF Example)**.**

$$\begin{aligned}
F_{\mathcal{C}}(x) &= \int_0^x \mathcal{C}(x)\mathrm{d}x && (17)\\
&= \int_0^x \left(w_L \mathsf{S}_0(x) + w_R \mathsf{S}_1(x)\right) \mathrm{d}x && (18)\\
&= w_L \int_0^x \mathsf{S}_0(x)\mathrm{d}x + w_R \int_0^x \mathsf{S}_1(x)\mathrm{d}x && (19)\\
&= \begin{cases} w_L \int_0^x \mathsf{S}_0(x) & \textit{if } x < b^0 \\ w_L + w_R \int_0^x \mathsf{S}_1(x) & \textit{else} \end{cases} && (20)
\end{aligned}$$

Note that here we leverage the facts that: $\int_0^x \mathsf{S}_0(x) = 1$ *iff* $x \geq b^0$, $\int_0^x \mathsf{S}_1(x)\mathrm{d}x = 0$ *iff* $x < a^1$*, recalling that* $\mathrm{supp}(\mathsf{S}_0(x)) = (a^0, b^0]$ *and* $\mathrm{supp}(\mathsf{S}_1(x)) = (a^1, b^1]$.

Let's consider the case $x < b^0$*:*

$$\begin{aligned}
F_{\mathcal{C}}(x) &= w_L \int_0^x \mathsf{S}_0(x)\mathrm{d}x && \textit{Iff } (x < b^0) && (21)\\
&= w_L \left(w_L^0 \mathsf{S}_{00}(x) + w_R^0 \mathsf{S}_{01}(x)\right) && && (22)\\
&= w_L \left(w_L^0 \int_0^x \mathsf{S}_{00}(x)\mathrm{d}x + w_R^0 \int_0^x \mathsf{S}_{01}(x)\mathrm{d}x\right) && && (23)\\
&= \begin{cases} w_L w_L^0 \int_0^x \mathsf{S}_{00}(x)\mathrm{d}x & \textit{if } x < b^{00} \\ w_L \left(w_L^0 + w_R^0 \int_0^x \mathsf{S}_{01}(x)\mathrm{d}x\right) & \textit{else} \end{cases} && && (24)
\end{aligned}$$

Now let's consider the right branch case, $x \geq b^{00}$*:*

$$F_{\mathcal{C}}(x) = w_L \left(w_L^0 + w_R^0 \int_0^x \mathsf{S}_{01}(x)\mathrm{d}x \right) \qquad \textit{iff } (b^{00} \leq x < b^0) \tag{25}$$

$$= w_L \left(w_L^0 + w_R^0 \int_0^x \left[w_L^{01} \mathsf{L}_{010}(x) + w_R^{01} \mathsf{L}_{011}(x) \right] \mathrm{d}x \right) \tag{26}$$

$$= w_L \left(w_L^0 + w_R^0 \left[w_L^{01} \int_0^x \mathsf{L}_{010}(x)\mathrm{d}x + w_R^{01} \int_0^x \mathsf{L}_{011}(x)\mathrm{d}x \right] \right) \tag{27}$$

$$= \begin{cases} w_L \left(w_L^0 + w_R^0 \left[w_L^{01} \int_0^x \mathsf{L}_{010}(x)\mathrm{d}x \right] \right) & \textit{if } x < b^{010} \\ w_L \left(w_L^0 + w_R^0 \left[w_L^{01} + w_R^{01} \int_0^x \mathsf{L}_{011}(x)\mathrm{d}x \right] \right) & \textit{else} \end{cases} \tag{28}$$

Considering the right branch case again, $x \geq b^{010}$:

$$F_{\mathcal{C}}(x) = w_L \left(w_L^0 + w_R^0 \left[w_L^{01} + w_R^{01} \int_0^x \mathsf{L}_{011}(x)\mathrm{d}x \right] \right) \qquad \textit{iff } b^{010} \leq x < b^0 \tag{29}$$

$$= w_L \left(w_L^0 + w_R^0 \left[w_L^{01} + w_R^{01} \left\{ \frac{x - a^{011}}{b^{011} - a^{011}} \right\} \right] \right) \tag{30}$$

◁

We can now invert this function by writing the equation w.r.t. u (with u denoting the CDF value $F_{\mathcal{C}}(x)$ for some x) instead of x.

Example B.6 (Inverse CDF Example)**.**

$$u = w_L \left(w_L^0 + w_R^0 \left[w_L^{01} + w_R^{01} \left\{ \frac{x - a^{011}}{b^{011} - a^{011}} \right\} \right] \right) \qquad \textit{iff } F_{\mathcal{C}}(b^{010}) \leq u < F_{\mathcal{C}}(b^0) \tag{31}$$

$$\frac{u}{w_L} = w_L^0 + w_R^0 \left(w_L^{01} + w_R^{01} \left[\frac{x - a^{011}}{b^{011} - a^{011}} \right] \right) \tag{32}$$

$$\frac{u}{w_L} - w_L^0 = w_R^0 \left(w_L^{01} + w_R^{01} \left[\frac{x - a^{011}}{b^{011} - a^{011}} \right] \right) \tag{33}$$

$$\frac{\frac{u}{w_L} - w_L^0}{w_R^0} = w_L^{01} + w_R^{01} \left(\frac{x - a^{011}}{b^{011} - a^{011}} \right) \tag{34}$$

$$\frac{\frac{u}{w_L} - w_L^0}{w_R^0} - w_L^{01} = w_R^{01} \left(\frac{x - a^{011}}{b^{011} - a^{011}} \right) \tag{35}$$

$$\frac{\frac{\frac{u}{w_L} - w_L^0}{w_R^0} - w_L^{01}}{w_R^{01}} = \frac{x - a^{011}}{b^{011} - a^{011}} \tag{36}$$

$$x = (b^{011} - a^{011}) \frac{\frac{\frac{u}{w_L} - w_L^0}{w_R^0} - w_L^{01}}{w_R^{01}} + a^{011} \tag{37}$$

Rewriting

$$x = (b^{011} - a^{011}) \left\{ \frac{1}{w_R^{01}} \left[\frac{1}{w_R^0} \left(\frac{1}{w_L} u - w_L^0 \right) - w_L^{01} \right] \right\} + a^{011} \tag{38}$$

◁

Observing the structure of the derivation, this can be generalized and written as the following set of recursive rules. Letting $F_{\mathcal{C}}^{-1}$ denote the inverse CDF of the circuit with $F_{\mathsf{N}_b}^{-1}$ denoting the inverse CDF of a given node,

Definition B.7 (Recursive Definition of the Inverse CDF)**.**

$$F^{-1}_{\mathsf{N}_{\boldsymbol{b}}}(u) = \begin{cases} F^{-1}_{\mathsf{S}_{\boldsymbol{b}}}(u) & \textit{if} \quad \mathsf{N} \in \mathcal{N}(\mathcal{T}) \\ F^{-1}_{\mathsf{L}_{\boldsymbol{b}}}(u) & \textit{if} \quad \mathsf{N} \in \mathcal{L}(\mathcal{T}) \end{cases} \tag{39}$$

where

$$F^{-1}_{\mathsf{S}_{\boldsymbol{b}}}(u) = \begin{cases} F^{-1}_{\mathsf{N}_{\boldsymbol{b}0}}\left(\frac{u}{w^{\boldsymbol{b}}_L}\right) & \textit{if} \quad u < w^{\boldsymbol{b}}_L \\ F^{-1}_{\mathsf{N}_{\boldsymbol{b}1}}\left(\frac{u - w^{\boldsymbol{b}}_L}{w^{\boldsymbol{b}}_R}\right) & \textit{otherwise} \end{cases} \tag{40}$$

and

$$F^{-1}_{\mathsf{L}_{\boldsymbol{b}}}(u) = a^{\boldsymbol{b}} + u(b^{\boldsymbol{b}} - a^{\boldsymbol{b}}) \tag{41}$$

Noting that this is simply the inverse-CDF of a uniform distribution. Further note how the conditions arise for the case of sum nodes. It is the case that $u < w^{\boldsymbol{b}}_L \equiv u < F^{-1}_{\mathsf{S}_{\boldsymbol{b}}}(b^{\boldsymbol{b}0})$. This is due to the fact that $F^{-1}_{\mathsf{S}_{\boldsymbol{b}}}(b^{\boldsymbol{b}0}) = w^{\boldsymbol{b}}_L F^{-1}_{\mathsf{N}_{\boldsymbol{b}0}} + w^{\boldsymbol{b}}_R F^{-1}_{\mathsf{N}_{\boldsymbol{b}1}}(b^{\boldsymbol{b}0})$, where $F^{-1}_{\mathsf{N}_{\boldsymbol{b}0}}(b^{\boldsymbol{b}0}) = 1$ and $F^{-1}_{\mathsf{N}_{\boldsymbol{b}1}}(b^{\boldsymbol{b}0}) = 0$.

B.3 ENTROPY DERIVATION

The general definition for the entropy of a probability distribution $p(x)$ is

$$\mathcal{H}\left(p(x)\right) = -\int_{x \in \mathcal{X}} p(x) \log(p(x)) \mathrm{d}x \tag{42}$$

Applying this to the definition of our model $\mathcal{C}$,

$$\begin{aligned} \mathcal{H}\left(\mathcal{C}(x)\right) &= -\int_{x \in \mathcal{X}} \mathcal{C}(x) \log(\mathcal{C}(x)) \mathrm{d}x && (43) \\ &= -\int_{x \in \mathcal{X}} \mathsf{S}(x) \log(\mathsf{S}(x)) \mathrm{d}x && (44) \\ &= -\int_{x \in \mathcal{X}} (w_L \mathsf{N}_0(x) + w_R \mathsf{N}_1(x)) \log(w_L \mathsf{N}_0(x) + w_R \mathsf{N}_1(x)) \mathrm{d}x && (45) \\ &= -w_L \int_{x \in \mathcal{X}} \mathsf{N}_0(x) \log(w_L \mathsf{N}_0(x) + w_R \mathsf{N}_1(x)) \mathrm{d}x && (46) \\ &\quad - w_R \int_{x \in \mathcal{X}} \mathsf{N}_1(x) \log(w_L \mathsf{N}_0(x) + w_R \mathsf{N}_1(x)) \mathrm{d}x \end{aligned}$$

Note that due to determinism of $\mathcal{C}$ and subsequently every N_ϵ in $\mathcal{C}$, the terms inside the logarithm can be simplified. In this case, $\mathrm{supp}(\mathsf{N}) = \mathcal{X}$. Then by our construction, $\mathrm{supp}(\mathsf{N}_0) = A_0$, $\mathrm{supp}(\mathsf{N}_1) = A_1$ where $A_0 \bigcup A_1 = \mathcal{X}$ and $A_0 \bigcap A_1 = \emptyset$. As such, the integrals and terms inside the logarithms simplify as follows:

$$\mathcal{H}\left(\mathcal{C}(x)\right) = -w_L \int_{x \in A_0} \mathsf{N}_0(x) \log(w_L \mathsf{N}_0(x)) \mathrm{d}x - w_R \int_{x \in A_1} \mathsf{N}_1(x) \log(w_R \mathsf{N}_1(x)) \mathrm{d}x \tag{47}$$

since $\mathsf{N}_0(x) = 0$ for all $x \in A_1$, and $\mathsf{N}_1(x) = 0$ for all $x \in A_0$. Simplifying further,

$$\mathcal{H}\left(\mathcal{C}(x)\right) = -w_L \int_{x \in A_0} \mathsf{N}_0(x)[\log(w_L) + \log(\mathsf{N}_0(x))]\mathrm{d}x \tag{48}$$
$$- w_R \int_{x \in A_1} \mathsf{N}_1(x)[\log(w_R) + \log(\mathsf{N}_1(x))]\mathrm{d}x$$
$$= -w_L \int_{x \in A_0} \mathsf{N}_0(x) \log(w_L)\mathrm{d}x - w_L \int_{x \in A_0} \mathsf{N}_0(x) \log(\mathsf{N}_0(x))\mathrm{d}x \tag{49}$$
$$- w_R \int_{x \in A_1} \mathsf{N}_1(x) \log(w_R)\mathrm{d}x - w_R \int_{x \in A_1} \mathsf{N}_1(x) \log(\mathsf{N}_1(x))\mathrm{d}x$$
$$= -w_L \log(w_L) \int_{x \in A_0} \mathsf{N}_0(x)\mathrm{d}x - w_L \int_{x \in A_0} \mathsf{N}_0(x) \log(\mathsf{N}_0(x))\mathrm{d}x \tag{50}$$
$$- w_R \log(w_R) \int_{x \in A_1} \mathsf{N}_1(x)\mathrm{d}x - w_R \int_{x \in A_1} \mathsf{N}_1(x) \log(\mathsf{N}_1(x))\mathrm{d}x$$

Since our circuit construction represents a normalized distribution, this means that for any $\mathsf{N} \in \mathcal{T}$, $\int_{x \in \mathrm{supp}(\mathsf{N})} \mathsf{N}(x) = 1$. This leads to the further simplification of integral terms,

$$\mathcal{H}\left(\mathcal{C}(x)\right) = -w_L \log(w_L) - w_L \int_{x \in A_0} \mathsf{N}_0(x) \log(\mathsf{N}_0(x))\mathrm{d}x \tag{51}$$
$$- w_R \log(w_R) - w_R \int_{x \in A_1} \mathsf{N}_1(x) \log(\mathsf{N}_1(x))\mathrm{d}x.$$

Looking at the remaining integral terms, observe that these correspond to the entropy formulas of the child nodes. Hence, the above can be rewritten as

$$\mathcal{H}\left(\mathcal{C}(x)\right) = -w_L \log(w_L) + w_L \mathcal{H}\left(\mathsf{N}_0(x)\right) - w_R \log(w_R) + w_R \mathcal{H}\left(\mathsf{N}_1(x)\right).$$

If N_0 and N_1 are sum nodes, then $\mathcal{H}(\mathsf{N}_0(x))$ and $\mathcal{H}(\mathsf{N}_0(x))$ can be derived using the same approach. If N_0 and N_1 are leaf nodes, then the entropy corresponds to the closed form solution for the entropy of a uniform distribution. Put into a recursive definition:

Definition B.8 (Recursive Definition of the Entropy)**.** *The entropy of a deterministic probabilistic circuit $\mathcal{C}$ is defined as*

$$\mathcal{H}\left(\mathcal{C}(x)\right) = \begin{cases} \mathcal{H}(\mathsf{S}_{\boldsymbol{b}}(x)) & \text{if } \mathsf{N} \in \mathcal{N}(\mathcal{T}) \\ \mathcal{H}(\mathsf{L}_{\boldsymbol{b}}(x)) & \text{if } \mathsf{N} \in \mathcal{L}(\mathcal{T}) \end{cases} \tag{52}$$

where N denotes the root node of $\mathcal{C}$. Furthermore,

$$\mathcal{H}\left(\mathsf{S}_{\boldsymbol{b}}(x)\right) = w_L^{\boldsymbol{b}} \log(w_L^{\boldsymbol{b}}) + w_L^{\boldsymbol{b}} \mathcal{H}\left(\mathsf{N}_{\boldsymbol{b}0}(x)\right) \tag{53}$$
$$- w_R^{\boldsymbol{b}} \log(w_R^{\boldsymbol{b}}) + w_R^{\boldsymbol{b}} \mathcal{H}\left(\mathsf{N}_{\boldsymbol{b}1}(x)\right)$$

and

$$\mathcal{H}\left(\mathsf{L}_{\boldsymbol{b}}(x)\right) = \log(b^{\boldsymbol{b}} - a^{\boldsymbol{b}}) \tag{54}$$

where $\mathsf{N}_{\boldsymbol{b}0}$ and $\mathsf{N}_{\boldsymbol{b}1}$ are the child nodes of $\mathsf{N}_{\boldsymbol{b}}$ (in the case that $\mathsf{N}_{\boldsymbol{b}}$ is a sum node S), and $a^{\boldsymbol{b}}, b^{\boldsymbol{b}}$ are the endpoints of the interval over which $\mathsf{N}_{\boldsymbol{b}}$ is defined (in the case that $\mathsf{N}_{\boldsymbol{b}}$ is a leaf node $\mathsf{L}_{\boldsymbol{b}}(x) = \frac{1}{b^{\boldsymbol{b}} - a^{\boldsymbol{b}}}$).

B.4 REVERSE KL DIVERGENCE CALCULATION

Let us define a density q and a density p. The reverse KL divergence of q from p is denoted as $\mathrm{KL}(q \,\|\, p)$, and defined as:

$$\mathrm{KL}(q \,\|\, p) = \int q(\boldsymbol{x}) \log \frac{p(\boldsymbol{x})}{q(\boldsymbol{x})} \mathrm{d}\boldsymbol{x} \tag{55}$$
$$= -\int q(\boldsymbol{x}) \log q(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} + \int q(\boldsymbol{x}) \log p(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x}. \tag{56}$$

Note that $-\int q(\boldsymbol{x}) \log q(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}$ is the entropy of distribution q, and will be denoted as $-\mathcal{H}(q)$:

$$\mathrm{KL}(q \,\|\, p) = -\int q(\boldsymbol{x}) \log p(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} - \mathcal{H}\left(q\right). \tag{57}$$

Note also that $\int q(\boldsymbol{x}) \log p(\boldsymbol{x}) \, \mathrm{d}$ is the expected value of the log-likelihood of p w.r.t. q:

$$\mathrm{KL}(q \,\|\, p) = -\mathbb{E}_{\boldsymbol{x}\sim q}\left[\log p(\boldsymbol{x})\right] - \mathcal{H}\left(q\right). \tag{58}$$

C EXPERIMENTAL DETAILS

C.1 2D DENSITIES

We present results for 2D non-Gaussian target distributions. In Fig. 11, we include additional results for typical benchmark target densities (mixture, Neal's funnel, two-modal Gaussian, ring, and banana) that we approximate with 4-bit/8-bit BitVI, which captures the overall density and cross-dependencies well.

C.2 MLP NEURAL NETWORK MODELS

The experiments with the *Bayesian-benchmarks* data sets used the following hyperparameters and setup:

- NN hyperparameters
  - Hidden layer size 16×16 for $D \leq 500$ and 32×32 for $D > 500$
  - LayerNorm [Ba et al., 2016] applied to hidden layers (pre-activation)
- PC hyperparameters
  - Weight representations used two integer bits, except for the 2-bit model, which used zero integer bits
  - Depth-based regularization for circuit parameters $\epsilon \, d^2$ with $\epsilon = 0.1$
  - Circuit weights were initialized from a beta distribution based on the height of the sum node in the circuit. The beta distribution α and β were set as 2^h where h is the height of the sum node in the circuit.
- Training hyperparameters
  - Adam optimizer with a learning rate of 0.001
  - Batch size of 32 for $D \leq 500$ and 128 for $D > 500$
  - 64 samples for computing the Monte Carlo approximation of the posterior log-joint
  - Early stopping based on the validation set ELBO loss after 2000 epochs
  - 5-fold cross-validation into train and test sets
  - Validation set split from the train set with 20% of the train set data

C.3 ABLATION STUDIES

Banana Chopping

- NN hyperparameters
  - LayerNorm [Ba et al., 2016] applied to hidden layers (pre-activation)
- PC hyperparameters
  - Weight representations used 10 bits with no integer bits. A sign bit and nine fractional bits.
  - Depth-based regularization for circuit parameters $\epsilon \, d^2$ with $\epsilon = 0.001$.
  - Circuit weights were initialized from a beta distribution based on the height of the sum node in the circuit. The beta distribution α and β were set as 2^h where h is the height of the sum node in the circuit.
- Training hyperparameters
  - Training set of 2048 points
  - Validation set of 512 points
  - Adam optimizer with a learning rate of 0.01
  - Batch size of 256

Table 3: Trade-off between NN model complexity (units in hidden layers) and bitstring depth (2–12 bits). Accuracy and expected calibration error (ECE) on the two moons data suggest that even low bit depth models perform well, and the dominating factor in expressivity is the number of units in the NN. See Table 2 in the main paper for the NLPD.

(a) Accuracy

	[4, 4]	[6, 6]	[8, 8]	[10, 10]	[12, 12]	[14, 14]	[16, 16]
2	0.856	0.85	0.852	0.87	0.868	0.888	0.89
3	0.854	0.857	0.903	0.865	0.897	0.909	0.906
4	0.852	0.879	0.88	0.884	0.904	0.898	0.909
5	0.86	0.877	0.854	0.895	0.901	0.909	0.913
6	0.863	0.861	0.853	0.887	0.895	0.91	0.906
7	0.882	0.883	0.888	0.899	0.896	0.909	0.905
8	0.874	0.877	0.909	0.884	0.886	0.909	0.904
9	0.864	0.873	0.873	0.877	0.897	0.914	0.909
10	0.87	0.862	0.886	0.884	0.912	0.899	0.909
12	0.852	0.888	0.863	0.863	0.898	0.895	0.908

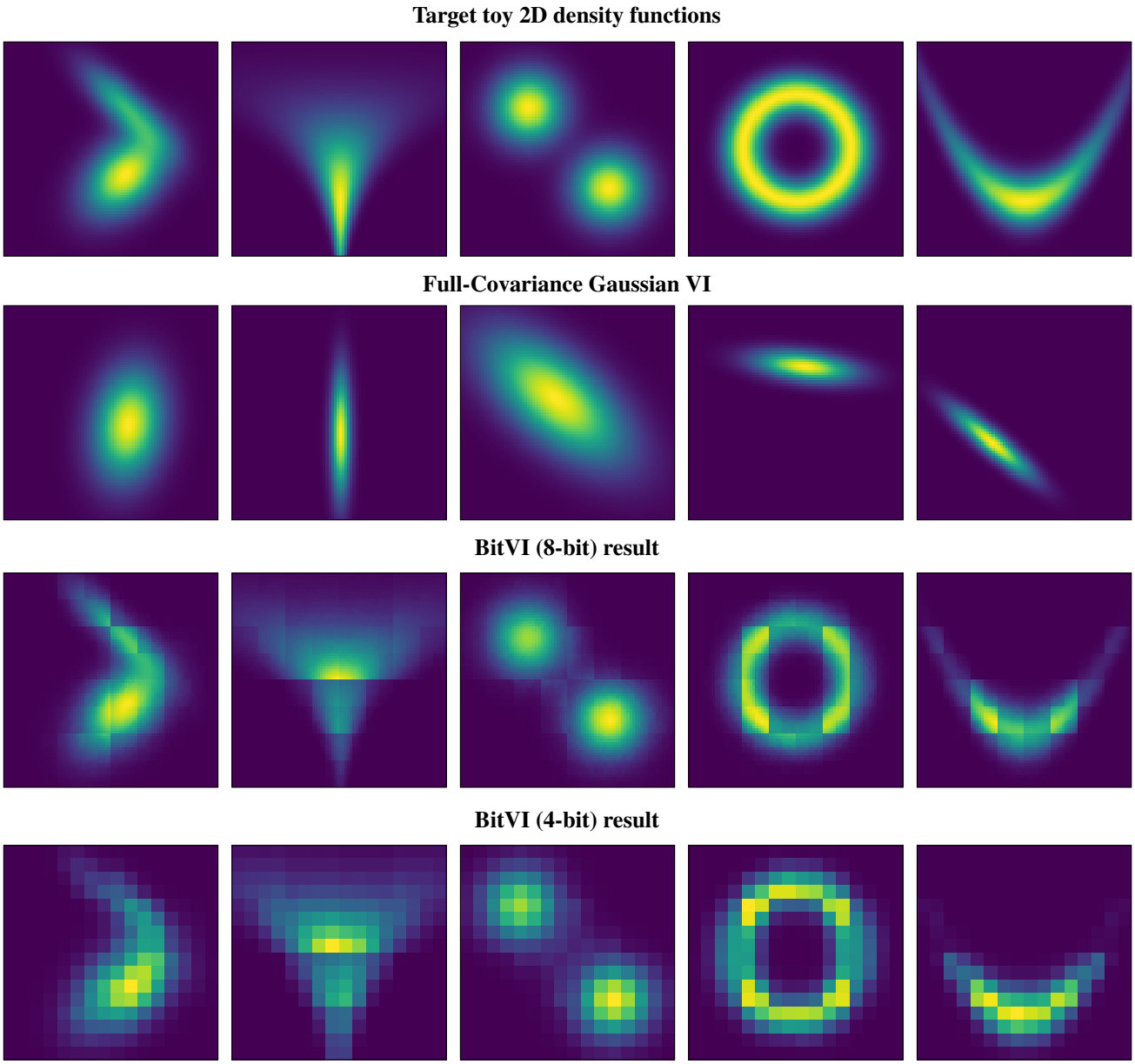
(b) ECE

	[4, 4]	[6, 6]	[8, 8]	[10, 10]	[12, 12]	[14, 14]	[16, 16]
2	0.059	0.053	0.061	0.064	0.065	0.055	0.06
3	0.06	0.071	0.051	0.053	0.046	0.042	0.049
4	0.064	0.064	0.055	0.045	0.042	0.045	0.038
5	0.061	0.057	0.058	0.053	0.042	0.043	0.044
6	0.06	0.067	0.057	0.048	0.044	0.039	0.046
7	0.063	0.053	0.048	0.046	0.041	0.042	0.046
8	0.061	0.051	0.038	0.047	0.048	0.045	0.042
9	0.055	0.056	0.053	0.053	0.045	0.04	0.042
10	0.057	0.058	0.056	0.05	0.037	0.047	0.046
12	0.064	0.05	0.055	0.053	0.041	0.05	0.045

D ADDITIONAL RESULTS

The following section contains additional results. In Fig. 11, we show results of applying BitVI in the task of approximating several benchmark non-Gaussian densities and compare the result to a baseline of a full-covariance Gaussian as the variational distribution. Furthermore, Table 3a and Table 3b provide further results (the accuracy and ECE metrics respectively) from the experiment investigating the relationship between NN model complexity and bitstring length.

D.1 ABLATION STUDIES



(a) Gaussian Mixture (b) Neal's Funnel (c) Two-modal Gaussian (d) Ring (e) Banana

Figure 11: 2D non-Gaussian target distributions. We include results for typical benchmark target densities (mixture, Neal's funnel, two-modal Gaussian, ring, and banana) that we approximate with 4-bit/8-bit BitVI, which captures the overall density and cross-dependencies well.