Stick-Breaking Bayesian Mixture Density Networks for Multimodal Probabilistic Regression

Leon Winheim*, **Markus Walker***, and **Uwe D. Hanebeck**
Intelligent Sensor-Actuator-Systems Laboratory (ISAS)
Institute for Anthropomatics and Robotics
Karlsruhe Institute of Technology (KIT), Germany
leon.winheim@kit.edu, markus.walker@kit.edu, uwe.hanebeck@kit.edu

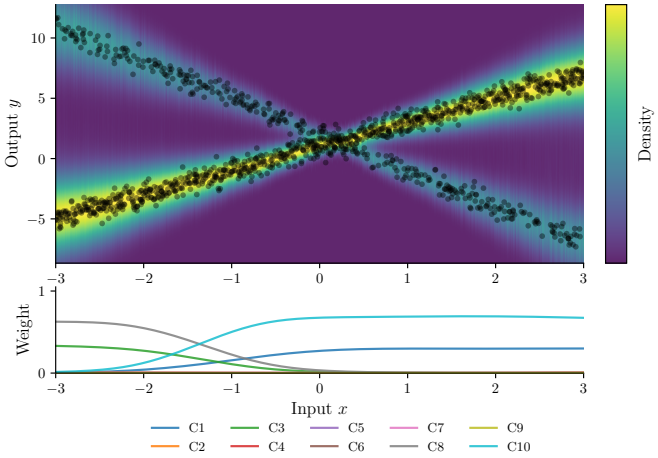
***Abstract*—Bayesian neural networks (BNNs) for probabilistic regression should possess the ability to represent input-varying multimodal data distributions. Typically, Gaussian likelihoods are assumed, which preclude this ability. Alternatively, mixture density networks (MDNs) are a well-suited representation. However, they require a reservoir with a fixed number of components with learned input-dependent weights. Current implementations exhibit mode-switching behavior for input-varying numbers of modes in the data. We propose incorporating principles from Dirichlet process mixtures (DPMs) into Bayesian MDNs using a stick-breaking construction. Our approach learns mixture weights through a Dirichlet-inspired mechanism that naturally favors using fewer components when appropriate. We present two architectural variants and evaluate them on synthetic multimodal regression tasks. The results indicate that our method achieves comparable predictive performance while showing improved behavior in terms of mode activation patterns and reduced mode-switching, suggesting that Dirichlet process (DP) principles can be successfully integrated into BNN likelihoods.**

***Index Terms*—Bayesian neural network, Dirichlet process mixture, multimodal regression.**

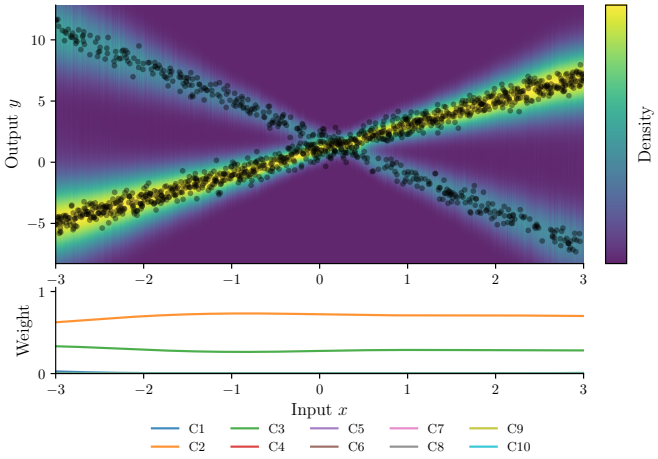
I. Introduction

Data-driven modeling techniques offer a powerful alternative to first-principles approaches for capturing complex input-output relationships. In probabilistic regression, the objective is to learn a model, e.g., a neural network (NN), that approximates the unknown conditional distribution of outputs given the inputs, based on observed data. A common assumption is that the likelihood function is Gaussian, which is often sufficient and considered standard practice [1]. However, training with unimodal likelihoods such as Gaussians can lead to suboptimal predictive distributions when the outputs in the data-generating process are multimodal (as in Fig. 1), e.g., resulting in averaging behavior where the true multimodal structure is not reproduced [2]. Therefore, we focus on the choice and parametrization of likelihoods, which directly affects the expressiveness of predictive distributions.

A classic approach to addressing this issue is to use mixture density networks (MDNs), which parameterize a mixture likelihood using the output of a neural network [3]. Extending these models to the Bayesian framework yields Bayesian mixture density networks (BMDNs), which additionally enable quantification of model uncertainty [4]. Importantly, incorporating Bayesian inference into NNs is orthogonal to the choice of likelihood parametrization: BNNs treat model parameters as random variables to capture epistemic uncertainty, while mixture-based likelihoods address multimodality in the data.



(a) BMDN (Crossed Lines data set)



(b) S2B (Crossed Lines data set)

Fig. 1: Comparison of Bayesian mixture density network (BMDN) and the proposed sigmoid stick-breaking (S2B) on the Crossed Lines Weighted data set.

*These authors contributed equally.

When the number of modes in the data set is known a priori, MDNs can be configured with the appropriate number of mixture components. However, if the number of modes is unknown, practitioners have to over-parametrize the model by specifying more mixture components than likely needed. This over-parametrization introduces two distinct issues: First, multiple model components may redundantly represent a single true mode at a given input, wasting model capacity. Second, and more critically, the model exhibits mode-switching behavior across the input space, where different components take turns representing the same true mode at different inputs rather than maintaining dedicated roles, as can be seen in Fig. 1a. This switching makes it challenging to identify and remove (prune) unused components after training, as all available components can appear active somewhere across the input space, even when the underlying data-generating process has only two or three modes. Consequently, the model remains unnecessarily large and complex.

To address this issue, we propose incorporating principles from Dirichlet process mixtures (DPMs) into the likelihood mechanism of these models. Our goal is to encourage sparse component activation while maintaining dedicated roles for active components across the input space. This behavior is commonly described through the *rich-get-richer* metaphor, meaning that components already representing data modes should receive further support during learning while unused components are suppressed. Specifically, we employ a stick-breaking construction [5] to determine mixture weights, which naturally implements the rich-get-richer mechanism. Unlike standard MDNs where mixture weights are learned directly without constraints on sparsity, DPMs are designed to handle an unknown number of modes by adaptively allocating weight mass (probability mass) to only the necessary components. This approach enables the model to automatically suppress unused components while maintaining dedicated roles for active ones, thus addressing both redundancy and mode-switching. To the best of our knowledge, the application of DP approaches to regression BNNs has not been explored in literature.

Contributions: We propose a novel architecture that combines Bayesian MDNs with Dirichlet process mixture (DPM) likelihoods to address probabilistic regression with potentially multimodal data. Our approach employs a stick-breaking construction to determine mixture weights, promoting sparse component activation and stable mode assignment (see Fig. 1b). We present two architectural variants and evaluate them on synthetic multimodal regression tasks, demonstrating improved mode activation patterns with comparable predictive performance to standard Bayesian MDNs.

Notation: In this paper, underlined letters, e.g., $\underline{x}$, denote vectors, boldface letters, such as $\underline{\boldsymbol{x}}$, represent random variables, while sets are represented as calligraphic letters, e.g., $\mathcal{D}$. Probability density functions are denoted by $f(\cdot)$.

II. Problem Description

We consider supervised learning for probabilistic regression tasks. Given a data set $\mathcal{D} = \{\underline{x}_i, \underline{y}_i\}_{i=1}^N$ consisting of N data points, with d_x-dimensional input vectors $\underline{x}_i \in \mathbb{R}^{d_x}$ and corresponding d_y-dimensional output vectors $\underline{y}_i \in \mathbb{R}^{d_y}$, our goal is to learn a model that approximates the true underlying conditional density $\hat{f}(\underline{y}|\underline{x})$ by $f(\underline{y}|\underline{x}, \underline{\theta})$. For a feedforward NN architecture $\underline{z} = g(\underline{x}, \underline{\theta})$, such as an multilayer perceptron (MLP), we treat the parameter vector $\underline{\theta}$ as a random vector $\underline{\boldsymbol{\theta}}$ to enable model parameter uncertainty quantification. This Bayesian approach, known as BNNs, captures epistemic uncertainty in the model parameters through the posterior distribution

$$f(\underline{\theta}|\mathcal{D}) = \frac{f(\mathcal{D}|\underline{\theta}) f(\underline{\theta})}{f(\mathcal{D})} \ , \tag{1}$$

where $f(\mathcal{D}|\underline{\theta})$ is the likelihood, $f(\underline{\theta})$ is the prior, and $f(\mathcal{D})$ is the marginal likelihood. The posterior $f(\underline{\theta}|\mathcal{D})$ represents updated beliefs about the parameters after observing data. Since exact inference is typically intractable, approximate methods such as variational inference (VI) [6] or Markov chain Monte Carlo (MCMC) [7] are employed [1].

To represent multimodal conditional distributions, we employ Gaussian mixture likelihoods instead of single Gaussian likelihoods, which are restricted to unimodal outputs. The mixture likelihood is given by

$$f(\underline{y}|\underline{x}, \underline{\theta}) = \sum_{k=1}^{K} w_k(\underline{x}, \underline{\theta}) \, \mathcal{N}\Big(\underline{y}; \underline{\mu}_k(\underline{x}, \underline{\theta}), \mathbf{C}_k(\underline{x}, \underline{\theta})\Big) \, , \tag{2}$$

where $w_k(\underline{x}, \underline{\theta})$ is the k-th component's mixture weight, $\underline{\mu}_k(\underline{x}, \underline{\theta})$ is the mean vector, and $\mathbf{C}_k(\underline{x}, \underline{\theta})$ is the covariance matrix. The network outputs $\underline{z} = g(\underline{x}, \underline{\theta})$ specify all mixture parameters by mapping inputs $\underline{x}$ and model parameters $\underline{\theta}$ to mixture parameters. Specifically, $\underline{z}$ concatenates the K mixture weights $\{w_k\}_{k=1}^K$, K mean vectors $\{\underline{\mu}_k\}_{k=1}^K$, and the parameters defining the K covariance matrices $\{\mathbf{C}_k\}_{k=1}^K$ (e.g., variance vectors for diagonal covariances or their Cholesky factors). Assuming independent and identically distributed data, the likelihood of the entire data set is given by

$$f(\mathcal{D}|\underline{\theta}) = \prod_{i=1}^{N} f(\underline{y}_i|\underline{x}_i, \underline{\theta}) \ . \tag{3}$$

Training proceeds by approximately solving (1).

III. Related Work

Approaches to probabilistic regression with multimodal outputs fall into two main categories: (i) set distance-based methods and (ii) continuous likelihood-based methods.

Set distance-based methods avoid distributional assumptions altogether by employing set distances as loss functions. Methods using Wasserstein distance [8] or modified Cramér–von Mises distance [2] enable models to reproduce arbitrary distribution shapes by comparing predictions and training data without parametric constraints.

Most closely related to our work are methods based on continuous mixture likelihoods as in (2). Classic MDNs learn the mixture parameters through maximum likelihood estimation, where a deterministic MLP outputs the component weights, means, and covariance matrices [3]. Bayesian MDNs extend this

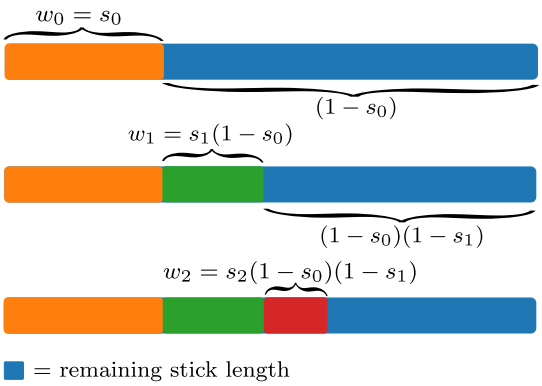


Fig. 2: Example of the stick-breaking construction; needs to be read top down, as the weights are constructed sequentially. Samples s_n are used to describe the proportion of the remaining stick length in every step that is then left over for the remaining mixture weights.

framework by treating network parameters as random variables, capturing both epistemic and aleatoric uncertainty through Bayesian inference, by approximately solving (1) [1], [4], [9]. These mixture-based approaches generalize the standard unimodal regression setting, where BNNs typically employ a single Gaussian likelihood [10], [11]. While computationally efficient and sufficient for many applications, the unimodal assumption limits the ability to represent multimodal conditional distributions, motivating the use of mixture likelihoods.

Beyond mixture-based regression, Dirichlet process mixtures (DPMs), i.e., models that adaptively determine the number of dominant mixture components, have been applied extensively in other domains. [12] demonstrated automatic clustering and classification for image data sets using DPM fitted by VI, though without neural network parameterization. Classification tasks with DPMs appear in [13], while Dirichlet-based approaches are explored in [14], [15]. In [14], a Dirichlet distribution is attached at the output of a MLPs, which may look similar but has a different goal of simplifying the computation of probabilities in classification and does not involve the DPs itself. The same is true for [15], where the Dirichlet distribution is only used to represent class probabilities, without any concept of the DP. Clustering with unknown cluster counts remains a classic DPM application [16].

Combinations of DPMs and NNs exist in different contexts. [17] uses DPMs as priors over network parameters (weights), rather than for modeling multimodal outputs. Variational autoencoders with DPM latent representations [18], [19] share similarities in DPM placement, but target generative modeling rather than direct prediction.

IV. Dirichlet Process Models

As mentioned in the previous section, the problem of an unknown number of mixture components (as encountered in clustering or classification) can be addressed by employing a DP. The DP is a fundamental tool in Bayesian nonparametric modeling that serves as a distribution over probability distributions. Rather than presenting the formal measure-theoretic definition, we adopt the equivalent constructive definition via the stick-breaking process, also known as the Sethuraman representation [5]. For details on the formal definition and properties of the DP, we refer to [5], [18].

The stick-breaking construction, introduced in [5], is an explicit, constructive representation of a Dirichlet Process $DP(\alpha, G_0)$, where α is the concentration parameter and G_0 is the base distribution. It shows that a draw $f(\omega) \sim \mathrm{DP}(\alpha, G_0)$ is almost surely a discrete probability mass function $f(\omega)$ composed of a countably infinite sum of point masses, given by

$$f(\omega) = \sum_{k=1}^{\infty} w_k \, \delta(\omega - \omega_k) \ , \tag{4}$$

where $\delta(\cdot)$ is a point mass at location ω_k, and the weights w_k are generated through a stick-breaking process. The base distribution G_0 serves as a prior from which the locations ω_k are drawn. The weights w_k can be constructed sequentially by drawing $s_k \sim \mathrm{Beta}(1, \alpha)$ and computing $w_k = s_k \prod_{j=1}^{k-1}(1 - s_j)$. This process ensures that the weights sum to one, i.e., $\sum_{k=1}^{\infty} w_k = 1$.

This allows the DP to flexibly allocate probability mass across an infinite number of components, while the concentration parameter α controls the expected number of components that receive significant weight. A smaller α leads to a more concentrated distribution with fewer active components, while a larger α results in a more diffuse distribution with many components receiving non-negligible weight. Intuitively, while α sets the global concentration tendency, each individual draw s_k dictates what fraction of the remaining stick component k keeps for itself versus passes on to subsequent components; a higher s_k claims more mass for component k and leaves a smaller fraction for all future weights. This sequential allocation creates the desired *rich-get-richer* effect: components that appear earlier in the sequence have access to more remaining probability mass, while later components can only claim from what remains. Consequently, earlier components tend to dominate unless their break proportions s_k are small, naturally encouraging sparse activation patterns where only a few components receive substantial weight.

The base distribution G_0 could be any distribution; e.g., in clustering, this could be a Gaussian distribution over cluster means and covariances. The choice of G_0 reflects prior beliefs about the component parameters and can significantly influence the resulting mixture model.

For a finite K-component mixture model, the stick-breaking process terminates at index K by enforcing $s_K = 1$, i.e.,

$$w_k = s_k \prod_{j=1}^{k-1}(1 - s_j) \quad \text{for } k = 1, \ldots, K \ , \tag{5}$$

with $w_1 = s_1$ and $s_k \in [0, 1]$. Since $s_K = 1$, the final weight $w_K = \prod_{j=1}^{K-1}(1 - s_j)$ absorbs the entire remainder of the stick, guaranteeing $\sum_{k=1}^{K} w_k = 1$ holds identically. An illustration of the stick-breaking construction is shown in Fig. 2.

V. Proposed Methods

We incorporate stick-breaking principles from Dirichlet process mixtures (DPMs) into MDNs to address the over-parametrization issues described in the introduction. Since standard feedforward networks have a predefined number of outputs, we build on the finite component stick-breaking construction from (5). This introduces an inductive bias toward sparse component activation through the rich-get-richer mechanism introduced above, while maintaining the flexibility to activate additional components as needed.

We present two variants for integrating this principle into Bayesian MDNs. First, we introduce the shared base network architecture, followed by the details of each variant.

A. Base Network Architecture

Both variants build upon the vanilla MDN architecture illustrated in Fig. 3a. The MDN outputs K mixture weights, K mean vectors $\underline{\mu}_k(\underline{x}, \underline{\theta})$, and K covariance matrices $\mathbf{C}_k(\underline{x}, \underline{\theta})$ for the mixture components. In the often-used simplified case of diagonal covariance matrices, $\mathbf{C}_k(\underline{x}, \underline{\theta})$ reduces to a variance vector $\underline{\sigma}_k^2(\underline{x}, \underline{\theta})$.

The network can use arbitrary activation functions in hidden layers, such as ReLU or GELU. In the output layer, mean vectors are unconstrained, making linear activations practical, while variance parameters use softplus activation to ensure positivity. For the mixture weights, standard MDNs typically employ a softmax activation to ensure normalized, positive weights. Our proposed variants differ in how these weights are generated through finite stick-breaking constructions introduced in Sec. IV, as detailed below.

B. Input-Dependent Concentration (IDC) Variant

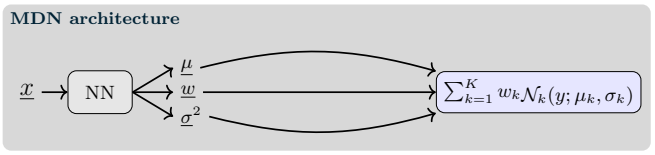
The first variant, illustrated in Fig. 3b, learns an input-dependent concentration (IDC) parameter $\alpha(\underline{x})$. For each input, the network outputs a scalar concentration value $\alpha(\underline{x}) > 0$ that parametrizes a Beta distribution $\text{Beta}(1, \alpha(\underline{x}))$ in the stick-breaking process. The break proportions are then sampled as

$$s_k \sim \text{Beta}\big(1, \alpha\left(\underline{x}\right)\big) \quad \text{for } k = 1, \ldots, K-1 \ , \tag{6}$$

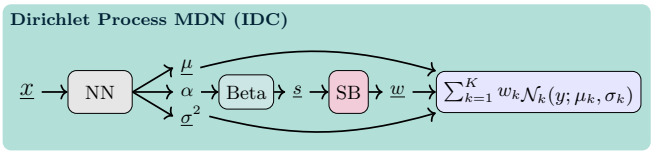
with $s_K = 1$. These samples are processed through the stick-breaking construction (5) to yield mixture weights.

Since we employ gradient-based approximate Bayesian inference methods such as VI for training, gradients must flow through the sampling operation. The reparameterization trick addresses this by expressing the sampling process as a deterministic transformation of parameter-independent random noise, thereby separating the randomness from the distribution parameters and allowing gradients to propagate through them. We apply the reparameterization trick for Beta distributions using the Kumaraswamy distribution as proposed by [18]. Specifically, we sample $\epsilon \sim \text{Kumaraswamy}(1, \alpha(\underline{x}))$ and transform it to obtain s_k, allowing gradients to propagate through $\alpha(\underline{x})$.

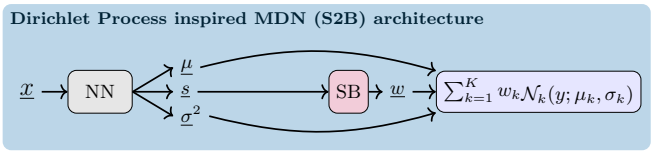
However, this approach introduces additional stochasticity during both forward and backward passes. This stochasticity may lead to unstable behavior in practice, which is why we propose a second version in the following subsection.



(a) Vanilla MDN architecture



(b) Input-dependent concentration (IDC) architecture



(c) Sigmoid stick-breaking (S2B) architecture

Fig. 3: Architectural comparison of the discussed approaches: (a) Vanilla MDNs directly output mixture weights, means, and variances; (b) IDC variant outputs an input-dependent concentration parameter $\alpha(\underline{x})$ that parametrizes a Beta distribution for the stick-breaking weight construction (SB); (c) sigmoid stick-breaking (S2B) variant directly predicts stick proportions s_k eliminating Beta sampling.

C. Sigmoid Stick-Breaking (S2B) Variant

To avoid the additional stochasticity introduced by sampling in the IDCs variant Sec. V-B, the second variant shown in Fig. 3c adopts a more direct approach. Instead of learning a concentration parameter that indirectly controls the break proportions through sampling, the network directly predicts $K-1$ break proportions $s_1, \ldots, s_{K-1}$, with $s_K = 1$ fixed as required by the finite stick-breaking construction in (5). The network outputs are constrained to $[0, 1]$ via sigmoid activation functions in the output layer

$$s_k = \sigma(z_k) = \frac{1}{1 + \exp(-z_k)} \quad \text{for } k = 1, \ldots, K-1 \ , \tag{7}$$

where z_k are the raw network layer outputs before the output layer. The deterministic stick-breaking mechanism then computes the mixture weights.

This approach eliminates the sampling step required in the IDC variant, resulting in more stable training while preserving the beneficial properties of stick-breaking weight allocation. The rich-get-richer mechanism remains intact: larger s_k values claim more probability mass for component k and leave less for subsequent components, naturally encouraging sparse mode activation patterns.

We name this specific approach "DP Inspired" rather than a strict DP implementation due to the simplification. The S2B variant specifically deviates from the strict DP framework by eliminating the Beta sampling entirely, instead predicting break proportions deterministically through sigmoid activations.

TABLE I: Model Comparison based on $\overline{\text{NLL}}$ as described in Sec. VI-B.

data set	Model	Neurons	Runs	$\overline{\text{NLL}} \pm \sigma$
Crossed Lines	S2B	10	50	1.3166 ± 0.0201
Crossed Lines	BMDN	10	50	1.3171 ± 0.0217
Crossed Lines	IDC	10	50	1.3835 ± 0.0522
Double Cubic	S2B	50	50	1.9313 ± 0.0391
Double Cubic	MDN	50	50	1.9391 ± 0.0447
Double Cubic	IDC	50	50	2.0035 ± 0.0612
Variable Modes	S2B	10	50	-0.1939 ± 0.0377
Variable Modes	BMDN	10	50	-0.1997 ± 0.0377
Variable Modes	IDC	10	50	0.0957 ± 0.084
Three Lines	S2B	10	50	1.6750 ± 0.0055
Three Lines	BMDN	10	50	1.6746 ± 0.0077
Three Lines	IDC	10	50	1.8063 ± 0.0416

D. Learning Setup

Both proposed variants parametrize the mixture likelihood in (2) and can be trained using standard Bayesian inference techniques such as VI or MCMC. Note that both variants are also compatible in non-Bayesian settings, where the network parameters are treated as deterministic and optimized via maximum likelihood estimation, since both variants are end-to-end differentiable. In this work, we focus on the Bayesian setting to capture both epistemic and aleatoric uncertainty.

VI. Experiments

To evaluate the influence of the stick-breaking construction, we compare our proposed S2B and IDC methods directly against standard BMDNs.

A. Data Sets and Evaluation Setup

We evaluate on four synthetic data sets designed to exhibit different multimodal characteristics. The "Crossed Lines" data set is generated as a two-component Gaussian mixture at every x according to

$$\begin{aligned} f(y \mid x) = 0.7\mathcal{N}(y\,;\, 2x+1, \sigma) \\ + 0.3\mathcal{N}(y\,;\, -3x+2, \sigma) \quad , \end{aligned} \tag{8}$$

with $\sigma = 0.5$. The "Double Cubic" data set is generated in a similar way according to

$$\begin{aligned} f(y \mid x) = 0.7\mathcal{N}\big(y\,;\, x^3, \sigma\big) \\ + 0.3\mathcal{N}\big(y\,;\, -x^3, \sigma\big) \quad , \end{aligned} \tag{9}$$

with $\sigma = 1.0$. The "Variable Modes" data set is generated following

$$f(y \mid x) = \begin{cases} \mathcal{N}(y; 0, \sigma^2), & x < -2, \\ \frac{1}{2} \sum\limits_{\mu \in \{0,2\}} \mathcal{N}(y; \mu, \sigma^2), & -2 \le x < 0, \\ \frac{1}{3} \sum\limits_{\mu \in \{0,2,4\}} \mathcal{N}(y; \mu, \sigma^2), & 0 \le x < 1, \\ \frac{1}{2} \sum\limits_{\mu \in \{0,3\}} \mathcal{N}(y; \mu, \sigma^2), & 1 \le x < 2, \\ \mathcal{N}(y; 0, \sigma^2), & x \ge 2, \end{cases} \tag{10}$$

with $\sigma = 0.1$ The "Three Lines" example is defined as

$$f(y \mid x) = \frac{1}{3} \sum_{\mu \in \{0,3,6\}} \mathcal{N}\big(y\,;\, \mu + 0.3x, \sigma^2\big)\,, \tag{11}$$

with $\sigma = 0.08$. We used 1250 random training points and 1250 random test points for each data set and model except for the "Three Lines" example, where we used 2500 each to ensure good coverage of the higher number of equally weighted modes.

All results are averaged over 50 runs with different random seeds. Representative results are shown in Fig. 1 and Figs. 4 to 6 and summarized in Tab. I. All models use a location mixture with one shared learnable output variance parameter. All inference is performed using stochastic variational inference [6] as implemented in NumPyro [18], with a mean-field Gaussian variational posterior. We train for 10 epochs with 1000 steps per epoch at a learning rate of 0.001. The network architecture consists of one hidden layer with either 10 or 50 neurons and ReLU activation.

B. Predictive Performance

We evaluate probabilistic prediction quality using the negative log-likelihood (NLL) of test observations under the predicted mixture distributions. We compute the NLL analytically from the predictive distribution.

Given S posterior samples $\{\underline{\theta}^{(s)}\}_{s=1}^{S}$ from the approximated weight posterior (1), the predictive density at test point (x_i, y_i) is given by

$$f(y_i|x_i) = \frac{1}{S} \sum_{s=1}^{S} \sum_{k=1}^{K} w_{i,k}^{(s)}\, \mathcal{N}(y_i; \mu_{i,k}^{(s)}, \sigma^{(s)}) \quad , \tag{12}$$

where $w_{i,k}^{(s)}$ are the predicted mixture weights, $\mu_{i,k}^{(s)}$ are the component means, and $\sigma^{(s)}$ the shared predictive standard deviation for posterior sample $\underline{\theta}^{(s)}$. For predictive performance comparison, we use the NLL over the entire test data given by

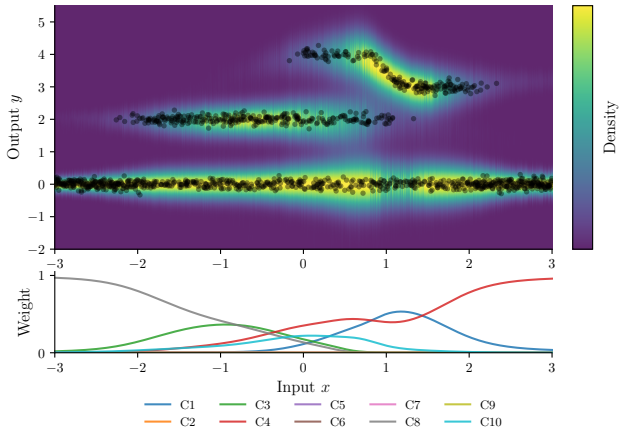
$$\overline{\text{NLL}} = -\frac{1}{N_{\text{Test}}} \sum_{i=1}^{N_{\text{Test}}} \log f(y_i|x_i) \quad , \tag{13}$$

where N_{Test} is the total number of test points.

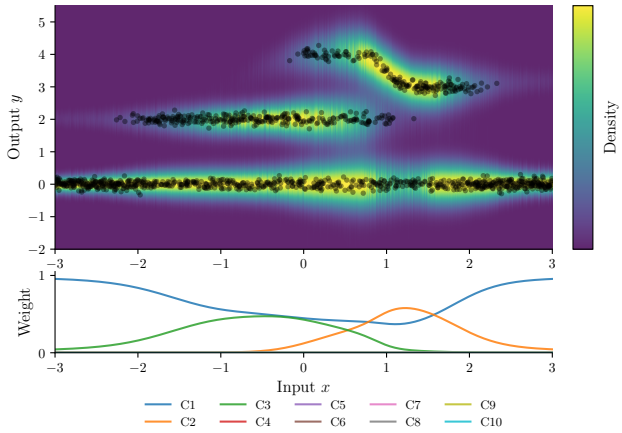
The results in Tab. I show that IDC achieves the highest (worst) NLL, attributable to the additional stochasticity from Beta-distribution sampling during inference. In contrast, the S2B variant matches the predictive performance of standard BMDN across all data sets. This demonstrates that the stick-breaking construction does not impair learning capacity despite imposing additional structure on the mixture weights.

C. Mode Utilization

Beyond predictive performance, we are particularly interested in mode activation behavior and the ability to prune unused components. Since IDC exhibits inferior predictive performance, we focus on comparing the proposed S2B variant to the BMDN. We define a mode as active if its mixture weight exceeds a threshold of 0.1. Desirable behavior is characterized by using the minimal number of components

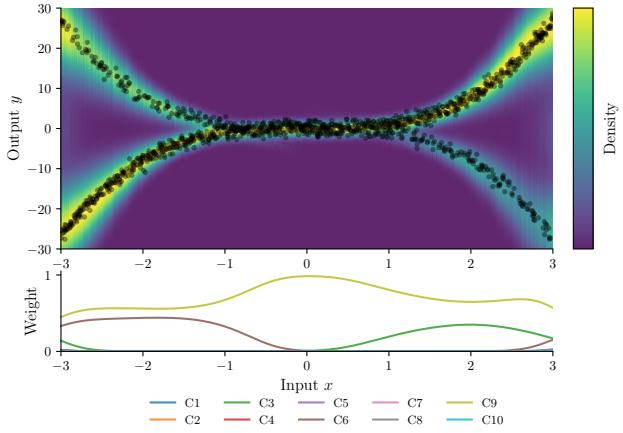


(a) BMDN (Variable Modes)

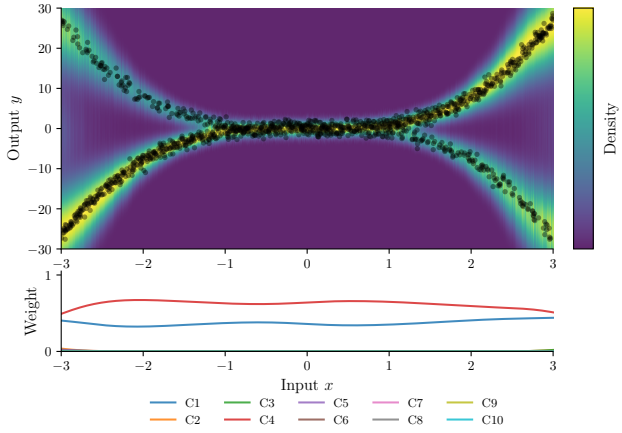


(b) S2B (Variable Modes)

Fig. 4: Comparison of BMDN (a) and the proposed S2B (b) on the Variable Modes data set.



(a) BMDN (Double Cubic)



(b) S2B (Double Cubic)

Fig. 5: Comparison of the BMDN (a) and the proposed S2B (b) on the Double Cubic data set.

necessary to represent the data-generating process, while maintaining stable component assignments across the input space (no mode-switching). For a bimodal data set, this means that two components should remain continuously active without switching roles. Such behavior would enable pruning of unused components after training, reducing model size.

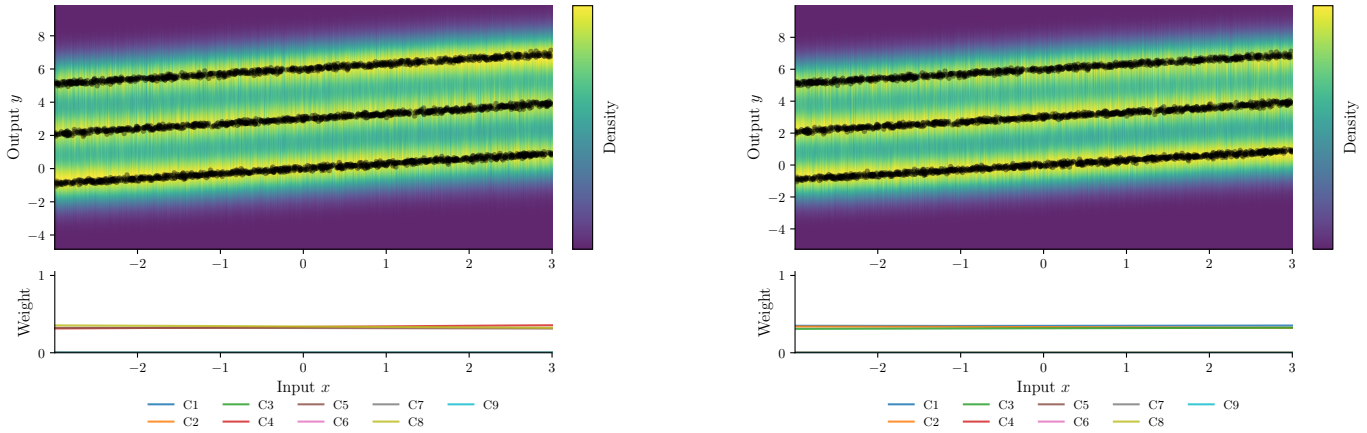
We visualize the mode activation behavior in Fig. 1a and Fig. 1b. The S2B result in Fig. 1b shows more desirable stable behavior, with a small number of components consistently representing the underlying modes. In contrast, the result in Fig. 1a illustrates that a model may achieve a similar average number of active modes while using a larger number of components that switch roles across the input space. Thus, simply counting the number of active modes at individual input points does not necessarily reflect efficient component utilization.

We further visualize the component usage in Fig. 7, which shows the activation frequency of the components over all test data and random seeds ordered by their mode rank. A rapid *decay after the correct number of modes* indicates that only a small number of components are used consistently, while a slower decay suggests that additional components contribute substantially across the input space. The S2B model exhibits faster decay for most data sets, indicating that it achieves comparable predictive performance while relying on fewer distinct components.

D. Discussion

The experimental results reveal several important insights about the proposed stick-breaking approach. First, both S2B and BMDN achieve comparable predictive performance in terms of NLL across all evaluated data sets, confirming that the stick-breaking construction does not impair the model's learning capacity despite imposing additional structure on the mixture weights.

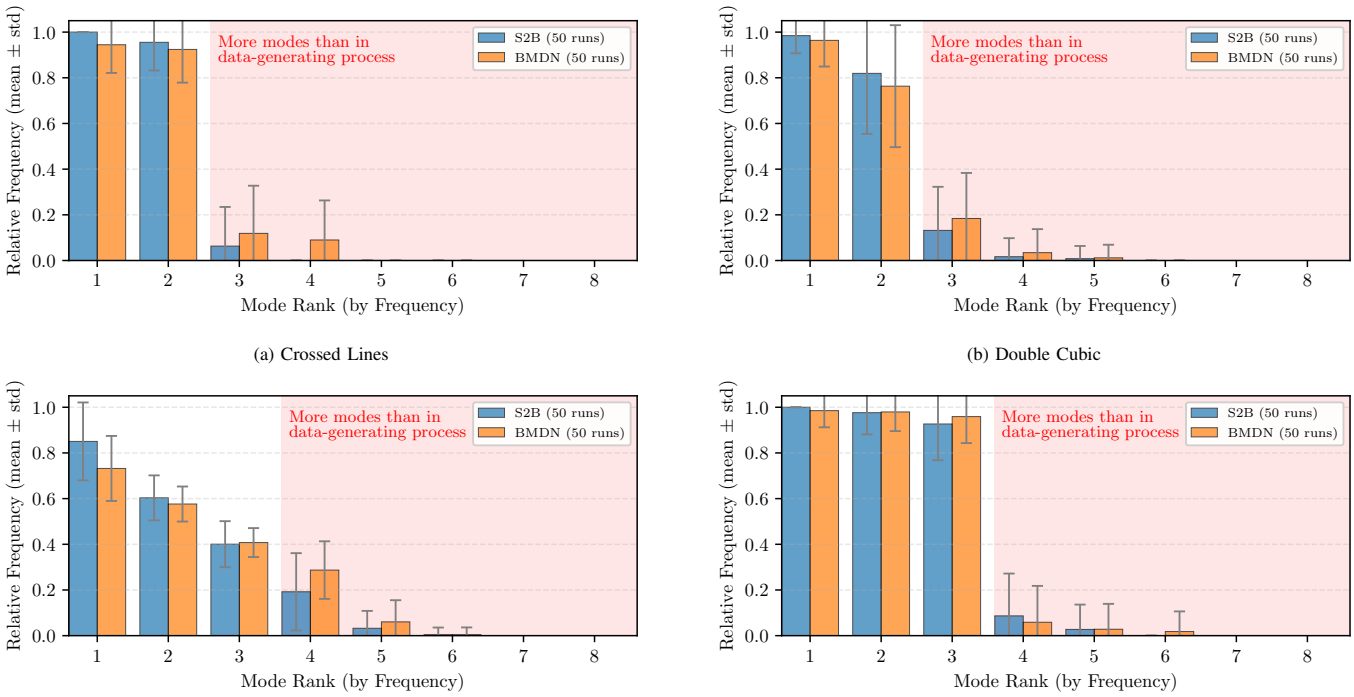
The key distinction lies in mode utilization robustness. The baseline BMDN can perform well in favorable conditions, as evidenced by the "Three Constants" data set where both models correctly activate only three modes. However, the S2B variant demonstrates more consistent behavior across all data sets and



(a) Bayesian MDN (Three lines)

(b) Stick Breaking MDN (Three lines)

Fig. 6: Comparison of Bayesian MDN and Stick Breaking MDN on the Three Lines data set.



(a) Crossed Lines

(b) Double Cubic

(c) Variable Modes

(d) Three Constants

Fig. 7: Visualization of the relative mode frequencies for specific runs: Reading the bars from right to left, note that the standard BMDN model uses more modes across the input space, which can only happen through mode-switching. In the proposed Dirichlet process-inspired version (S2B), fewer additional modes are activated, demonstrating more stable and sparse component utilization.

random seeds. The frequency analysis in Fig. 7 shows that S2B exhibits faster decay in component usage after the true number of modes, indicating that fewer distinct components are needed to represent the data-generating process. This suggests that the rich-get-richer inductive bias effectively promotes sparse and stable component activation patterns even when the baseline model occasionally achieves similar results.

However, the stick-breaking construction is not a perfect solution. While it improves mode utilization compared to standard BMDN, the activation frequency analysis in Fig. 7 reveals that some models still activate more components than present in the data-generating process. Nevertheless, the S2B approach shows fewer such cases compared to the baseline, indicating that the rich-get-richer mechanism effectively reduces redundant component usage, though it does not completely eliminate it.

The maximum number of modes is kept constant at 10 in our experiments. While we did not conduct a comprehensive analysis of the influence of this number, we expect the proposed methods to also mitigate the potential negative effects of a

strong mismatch between the actual number of modes and the maximum number of modes the model can accommodate, due to rich-get-richer dynamics.

From a practical perspective, the S2B approach offers two specific advantages: (1) reduced mode-switching enables more interpretable component assignments across the input space, and (2) the consistent sparsity pattern facilitates post-hoc pruning of unused components, potentially reducing model size without explicit regularization. These properties make the method particularly suitable for applications where model interpretability and efficient resource allocation are priorities.

VII. Conclusions

We present a Dirichlet process-inspired approach for Bayesian mixture density networks (BMDNs) that incorporates stick-breaking principles to determine mixture weights. Two architectural variants were proposed: an input-dependent concentration method and a direct sigmoid stick-breaking (S2B) approach. The S2B variant demonstrated more robust training behavior while preserving the beneficial properties of the stick-breaking construction.

Our evaluation on synthetic multimodal regression tasks shows that the proposed method achieves comparable predictive performance to standard BMDNs while exhibiting improved mode activation patterns. Specifically, the S2B model uses fewer distinct components across the input space, enabling mixture component pruning after training.

Future research directions include investigating improved initialization strategies to further enhance the robustness of the stick-breaking construction, extending the method to multi-dimensional outputs, and exploring whether learning the concentration parameter can be made more stable for adaptive complexity control. Additionally, the influence of the maximum possible number of modes on prediction quality could be analyzed.

References

[1] Ethan Goan and Clinton Fookes. "Bayesian neural networks: An introduction and survey". In: *Case Studies in Applied Bayesian Data Science: CIRM Jean-Morlet Chair, Fall 2018* (2020).

[2] Leon Winheim, Hayk Amirkhanian, Markus Walker, Uwe D. Hanebeck, and Marco Huber. "Probabilistic regression via neural network ensembles". In: *Proceedings of the 29th International Conference on Information Fusion (FUSION 2026)*. Trondheim, Norway, 2026.

[3] Christopher M Bishop. *Mixture Density Networks*. Aston University, 1994.

[4] Alessandro Brusaferri, Matteo Matteucci, Stefano Spinelli, and Andrea Vitali. "Probabilistic electric load forecasting through Bayesian mixture density networks". In: *Applied Energy* 309 (2022).

[5] Jayaram Sethuraman. "A constructive definition of Dirichlet priors". In: *Statistica Sinica* 4.2 (1994).

[6] Matthew D. Hoffman, David M. Blei, Chong Wang, and John Paisley. "Stochastic variational inference". In: *Journal of Machine Learning Research* 14.1 (2013).

[7] Matthew D. Hoffman and Andrew Gelman. "The no-u-turn sampler: adaptively setting path lengths in Hamiltonian Monte Carlo". In: *The Journal of Machine Learning Research* 15.1 (2014).

[8] Cyril Benezet, Ziteng Cheng, and Sebastian Jaimungal. "Learning conditional distributions on continuous spaces". In: *Journal of Machine Learning Research* 26.105 (2025).

[9] Laurent Valentin Jospin, Wray Buntine, Farid Boussaid, Hamid Laga, and Mohammed Bennamoun. "Hands-on Bayesian neural networks – a tutorial for deep learning users". In: *IEEE Computational Intelligence Magazine* 17.2 (2022).

[10] Hayk Amirkhanian, Markus Walker, Uwe D. Hanebeck, and Marco F. Huber. "Bridging Bayesian inference and neural network training: equivalence of KBNN and statistical linearization". In: *Proceedings of the 28th International Conference on Information Fusion (FUSION 2025)*. 2025.

[11] Markus Walker, Hayk Amirkhanian, Marco F. Huber, and Uwe D. Hanebeck. "Trustworthy Bayesian perceptrons". In: *Proceedings of the 27th International Conference on Information Fusion (FUSION 2024)*. Venice, Italy, 2024.

[12] David M. Blei and Michael I. Jordan. "Variational inference for Dirichlet process mixtures". In: *Bayesian Analysis* 1.1 (2006).

[13] Qingyang Wu, He Li, Lexin Li, and Zhou Yu. "Quantifying intrinsic uncertainty in classification via deep Dirichlet mixture networks". In: *arXiv preprint: 1906.04450* (2019).

[14] Marius Hobbhahn, Agustinus Kristiadi, and Philipp Hennig. "Fast predictive uncertainty for classification with Bayesian deep networks". In: The 38th Conference on Uncertainty in Artificial Intelligence.

[15] Rouaa Hoblos, Noura Dridi, Noureddine Zerhouni, and Zeina Al Masry. "Dirichlet-based Monte Carlo dropout for uncertainty estimation in neural networks". In: *56es Journées de Statistique de La SFdS*. 2025.

[16] Naiqi Li, Wenjie Li, Yong Jiang, and Shu-Tao Xia. "Deep Dirichlet process mixture models". In: *Uncertainty in Artificial Intelligence*. PMLR, 2022.

[17] Wolfgang Roth and Franz Pernkopf. "Bayesian neural networks with weight sharing using Dirichlet processes". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42.1 (2020).

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

[19] Amine Echraibi, Joachim Flocon-Cholet, Stéphane Gosselin, and Sandrine Vaton. "On the variational posterior of Dirichlet process deep latent Gaussian mixture models". In: *ICML Workshop on Invertible Neural Networks, Normalizing Flows, and Explicit Likelihood Models*. 2020.