DSS-LVR FLOW: A UNIFIED METHOD FOR STRUCTURAL SPARSITY IN BAYESIAN NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Bayesian neural networks (BNNs) provide uncertainty-aware modeling of complex nonlinear relationships in scientific applications. Their flexibility, however, comes with rapidly expanding and redundant parameterizations, making it important to identify which predictors, hidden units, and larger computational structures are actually supported by the data. Discrete spike-and-slab (DSS) priors provide a natural Bayesian mechanism for this purpose: the point-mass spike enables exact exclusion, while the continuous slab retains posterior uncertainty for active parameters. We introduce DSS-LVR Flow, a flow-based method built on a DSS latent-variable representation (DSS-LVR), in which a group-specific activation latent variable and a shared continuous threshold jointly determine each inclusion decision. By changing which parameters share an inclusion state, the same construction supports predictor- and hidden-unit-level selection, while combinations of these primitive states induce edge and path activity without additional structural latent variables. DSS-LVR Flow uses a role-aware inverse autoregressive flow (IAF) to approximate the joint posterior. It also introduces a smooth, bounded, exact-zero structural activation function with a controllable transition region for gradient-based optimization. Experiments show that DSS-LVR Flow efficiently approximates parameter posteriors with accuracy close to that of long-run MCMC. In multilayer perceptrons, it also supports multiple structural sparsity targets, producing highly sparse structures while preserving accurate nonlinear function recovery.

1 INTRODUCTION

Bayesian neural networks (BNNs) extend traditional Bayesian parametric models by allowing predictor–outcome relationships to be modeled through flexible nonlinear functions and interactions rather than through simple parametric models. By placing probability distributions over neural-network parameters, BNNs quantify uncertainty in these nonlinear relationships rather than returning a single fitted function. This flexibility, however, comes at the cost of over-parameterization, motivating the need for sparsity and regularization. Discrete spike-and-slab (DSS) (Ishwaran & Rao, 2005) priors provide a natural Bayesian mechanism for structural selection: a point-mass spike represents exact exclusion, while a continuous slab quantifies posterior uncertainty for non-zero active parameters. In BNNs, these two components can be applied to introduce additional structure in DNNs.

Recent work in BNN methods therefore focuses on DSS prior models that introduce sparsity at different structural scales. For instance, Latent Binary Bayesian Neural Networks (LBBNNs) use individual parameter-wise DSS priors and variational inference for posterior inference (Skaaret-Lund et al., 2024; Høyheim et al., 2026).

Structured spike-and-slab BNNs instead remove complete hidden units, directly targeting node-level sparsity (Jantre et al., 2025), whereas weight-sharing BNNs assign a common inclusion probability to all first-layer weights associated with one predictor and thereby target nonlinear predictor selection (Mishra et al., 2026). Together, these approaches demonstrate the relevance of predictor-, connection-, unit-, and path-level uncertainty, but typically introduce them through structure-specific parameterizations.

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

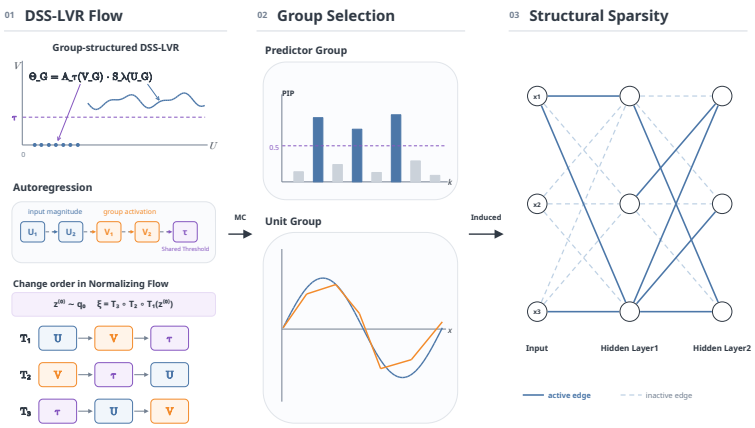


Figure 1: Overview of the proposed framework.

These developments motivate a general principle for defining structured DSS prior distributions for BNNs.

We introduce a flexible and computationally scalable latent-variable representation of group-structured DSS priors, which we refer to as DSS-LVR. Parameters of the BNN are organized into different structural groups (for instance based on sets of layers and/or neurons). Individual parameters of the BNN are then represented through a product of an individual slab and a set of group-specific activation components. These components are in turn defined as the push-forward of continuous random variables under basic slab and activation maps.

Posterior inference in the augmented-variable DSS representation enables flexible variational approximations beyond traditional mean-field variational inference (Skaaret-Lund et al., 2024; Mishra et al., 2026).

We utilize flexible inverse autoregressive flows (IAF) to approximate the joint posterior over the latent-variable space of our augmented DSS-LVR representation. Posterior distributions for individual parameters or structural quantities are obtained as marginals or functionals of draws from this joint approximation rather than being modeled independently as in mean-field variational inference.

Our IAF flow approximation represents each flow layer by an autoregressive triangular transformation whose affine parameters are produced by a masked network (Kingma et al., 2016). Compared with standard affine coupling flows, all affine parameters in an IAF layer are computed in an autoregressive fashion.

We evaluate our DSS-LVR prior and the IAF flow in a series of increasingly complex experiments, from shallow networks to sparse deep neural networks. We show in the sparse setting that our IAF flow recovers the gold-standard MCMC-based posterior distribution well.

Our experiments show that in sparse predictor settings, the DSS-LVR prior with IAF flow is competitive with and in many instances superior to alternative state-of-the-art BNN models in variable selection performance and functional recovery.

2 Method

We consider a BNN setup, where n data points $\mathcal{D}_n = \{(Y_i, X_i)\}_{i=1}^n$ are generated independently according to probability model $Y_i \mid X_i = x, \theta \sim p(Y_i \mid f_\theta(x))$ where $f_\theta(\cdot)$ is a deep neural network (DNN) parameterized by a collection of weights and bias parameters collected in $\theta \in \Theta \subset \mathbb{R}^K$. This setting includes DNN architectures for classification $Y_i \in \mathcal{Y} = \{1, 2, \ldots, C\}$ where $x \to f_\theta(x) = (f_{\theta,1}(x), \ldots, f_{\theta,C}(x))$ indicates class predictions $p[Y_i = c \mid X_i = x] = \frac{\exp\{f_{\theta,c}(x)\}}{\sum_{c'} \exp\{f_{\theta,c'}(x)\}}$ or regression problems when $Y \in \mathbb{R}$ where $f_\theta(x) = E[Y_i \mid X_i = x]$ indicates the mean function.

We focus on high-dimensional predictor settings where $X_i = (X_{i,1}, \ldots, X_{i,p})' \in \mathbb{R}^p$ and p is large, but only a small number of set of these predictors $X_{i,j}$ have meaningful effects on the outcome.

We seek to estimate the DNN f_θ and predict $Y_i \mid X_i$ using the Bayesian approach. Following the Bayesian setting we place a prior distribution $\pi(\theta)$ on the unknown θ parameters of the DNN that encodes the prior sparsity assumption. We then summarize information on the DNN f_θ through the likelihood $L(\theta; \mathcal{D}_n) = \prod_{i=1}^n p(Y_i \mid f_\theta(X_i))$ and update our prior sparsity guess into the posterior evidence $\pi(\theta \mid \mathcal{D}_n) \propto \pi(\theta) L(\theta; \mathcal{D}_n)$ and predict future observation using the corresponding posterior predictive distribution $f(y|x, \mathcal{D}_n) = \int p(y \mid f_\theta(x)) d\pi(\theta \mid \mathcal{D}_n)$.

2.1 Group-Structured Prior for Sparse BNNs

Shallow Neural Network. We introduce a computationally scalable, flexible sparse group-structures prior for DNNs. To simplify the presentation, we initially outline our prior distribution for a shallow network with a single-hidden-layer,

$$f_\theta(x) = b^{(2)} + \sum_{j=1}^{H} W_j^{(2)} \, \sigma\left((W_{j\cdot}^{(1)})^\top x + b_j^{(1)}\right) \quad \text{for } x = (x_1, \cdots, x_p) \in \mathbb{R}^p, \tag{1}$$

with $H \geq 1$ hidden units, activation function $\sigma(\cdot)$, hidden layer bias $b^{(1)} = (b_1, \cdots, b_H)'$, and hidden and top-layer weights $W^{(2)} \in \mathbb{R}^H$ and $W^{(1)} \in \mathbb{R}^{H \times p}$, i.e. $\theta = \{b^{(1)}, b^{(2)}, W^{(1)}, W^{(2)}\}$. If p and H are large, there is a strong rational to regularize and shrink the effective number of hidden unites and covariates x_k that predict the outcome Y_i. If only a small subset of predictors impact the outcome (i.e. some of the columns of $W^{(1)}$ are parse), group priors such that the Group-lasso

$$\pi(W_{\cdot k}^{(1)}) \propto \exp\left\{-\lambda_x ||W_{\cdot k}^{(1)}||_2\right\}$$

or discrete spike-and-slab (DSS) prior distributions of the form

$$\pi(W_{\cdot k}^{(1)}) = (1 - q_0)\, \delta_0(W_{\cdot k}^{(1)}) + q_0 \, \pi_S(W_{\cdot k}^{(1)}) \quad \text{for } k = 1, \ldots, p, \tag{2}$$

can be utilized for structures regularization. Here, $\delta_0(\cdot)$ indicates the point mass at the zero vector, $q_0 \in [0, 1]$ is the prior probability that $||W_{\cdot k}^{(1)}||$ is non-zero (active), and $\pi_S(\cdot)$ is a slab prior distribution for the active predictor effects $W_{\cdot k}^{(1)}$. The parameter q_0 controls the sparsity of the vector $W_{\cdot k}^{(1)}$, and can be learned by specifying a hyperprior $q_0 \sim \pi_{q_0}(q_0)$.

Latent-variables DSS structured group prior representation. Posterior inference under the above DSS prior is computationally demanding. The aim is to conduct (approximate) posterior inference for sparse DNNs without having to resort to time-consuming MCMC or crude deterministic variational approximations. Our approach introduces a novel latent variable parameterization for grouped-structured DSS prior generalizing equation 2 via an augmented latent variable representation. Let $G_1, \cdots, G_g$, with $G_j \subset \{1, \ldots, K\}$ for $j = 1, \ldots, g$, indicate a collection of group structures, i.e. sets of indices of θ. The sets can be overlapping and, $\cup_j G_j$ can generally differ from $\{1, \ldots, K\}$. Then we define our general group structures DSS prior for DNNs as

$$\begin{aligned} &\theta_j = S_{\lambda_{k_j}}(U_j) \times \prod_{\ell : j \in G_\ell} A_{\tau_{r_\ell}}(V_\ell) \quad \text{with} \\ &U_j \overset{iid}{\sim} \pi_U \quad \text{for} \quad j = 1, \ldots, K, \quad V_\ell \overset{iid}{\sim} \pi_V \text{ for } \ell = 1, \ldots, g \quad \text{and} \quad \tau_r \overset{iid}{\sim} \pi_\tau, \end{aligned} \tag{3}$$

where π_U, π_V and π_τ are absolutely continuous distributions on the real line, the activation function $\mathcal{A}_\tau : \mathbb{R} \to [0, +\infty)$ determines the sparsity of each individual parameter θ_j, and the slab function $S_\tau : \mathbb{R} \to \mathbb{R}$ controls the magnitude and tail behavior of the non-zero components $\theta_j \neq 0$ of θ. By convention, if $j \notin G_\ell$ for any group, then the above product $\prod_{\ell : j \in G_\ell} A_{\tau_{r_\ell}}(V_\ell) = 1$ is set to one. The parameters $\tau_c \in \mathbb{R}, c = 1, \ldots, g$ with prior distribution $\tau_c \sim \pi_\tau$ is shared by multiple groups, i.e. all groups G_ℓ such that $r_\ell = c$. It control the sparsity level – i.e. the average number of non-zero components – of θ-components θ_j belonging to a groups of variable $r_\ell = c$. Lastly the $\lambda_c > 0$ parameters controls the spread of the slab distribution, which can also be shared by multiple groups.

For the above shallow neural network and each covariate $k = 1, \ldots, p$, let $G_k = \{(1, 1, k), \ldots, (1, H, k)\}$ indicate the index of the k-th column $W_{\cdot k}^{(1)}$ of $W^{(1)}$. Moreover for each

hidden unit $j = 1, \ldots, H$, let $G_{p+j} = \{(1, j, 1), \ldots, (1, j, p)\}$ indicate the j-th row of $W^{(1)}$. Then the covariate and unit level DSS sparsity prior can be written as

$$W_{jk}^{(1)} = \mathcal{S}_{\lambda_1}\left(U_{jk}^{(1)}\right) \times \overbrace{\mathcal{A}_{\tau_1}\left(V_j^{(1)}\right)}^{ULS} \overbrace{\mathcal{A}_{\tau_x}\left(V_{xk}^{(1)}\right)}^{PLS} \tag{4}$$

$$b_j^{(1)} = \mathcal{S}_{\lambda_m}\left(U_j^{(1)}\right) \times \overbrace{\mathcal{A}_{\tau_m}\left(V_j^{(1)}\right)}^{ULS} \tag{5}$$

$$W_r^{(2)} = \mathcal{S}_{\lambda_2}\left(U_r^{(2)}\right) \tag{6}$$

$$b_j^{(2)} = \mathcal{S}_{\lambda_2}\left(U_j^{(2)}\right) \tag{7}$$

$$\begin{aligned} U_j^{(m)}, U_{jk}^{(m)} &\overset{iid}{\sim} \pi_U \qquad \text{for } k = 1, \ldots, H, \\ V_j^{(m)} &\overset{iid}{\sim} \pi_V \qquad \text{for } j = 1, \ldots, p_m \text{ and } m = 1, \ldots, 2 \\ V_{xk}^{(1)} &\overset{iid}{\sim} \pi_V \qquad \text{for } k = 1, \ldots, H. \end{aligned}$$

Example 1. As a simple example of the a grouped DSS prior we consider the indicator activation function in combination with an affine slab map

$$A_\tau(v) = I(v > \tau) \text{ and } S_\lambda(u) = \lambda u \text{ and } \pi_V = \pi_U = \pi_\tau = N(0, 1).$$

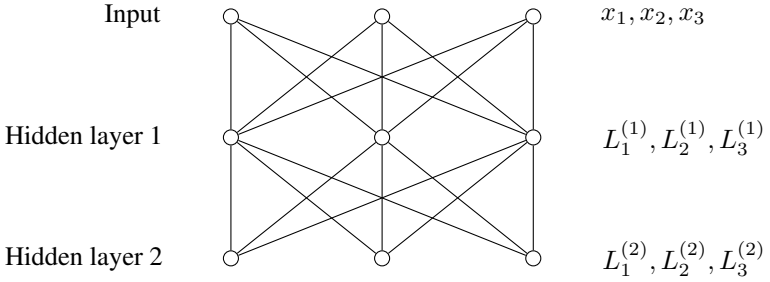
This model corresponds to a grouped DSS prior equation 3 with normal slab distribution and prior inclusion probability $\pi(W_{jk}^{(1)} \neq 0) = (1 - \Phi(\tau_1))(1 - \Phi(\tau_x))$.

Example 2. We can alternatively specify a more heavy-tailed grouped DSS prior by specifying a ReLU activation in combination with a polynomial slab map

$$A_\tau(v) = \max(v - \tau, 0) \text{ and } S_\lambda(u) = \lambda \text{sign}(u)|u|^k \text{ and } \pi_V = \pi_U = \pi_\tau = N(0, 1).$$

for $k \in \mathbb{N}$, where $\text{sign}(u)$ indicates the sign function. This prior model corresponds to a grouped DSS prior equation 3 with inclusion probability $(1 - \Phi(\tau_1))(1 - \Phi(\tau_x))$ for $W_{jk}^{(1)}$. Ween $k = 1$, the slab distribution behavior like a Bayesian LASSO (Laplace) distribution.

Multilayer Perceptrons. We now extend the prior construction for shallow neural networks to the general DNNs. Consider the usual multilayer perceptron architecture with $M > 1$ hidden layers and d_m neurons at hidden layer $m = 1, \ldots, M$.



Formally, we define the DNN using the usual recursion

$$\begin{aligned} x \to L^{(1)} &= \sigma(W^{(1)}x + b^{(1)}) \\ L^{(2)} &= \sigma(W^{(2)}L^{(1)} + b^{(2)}) \\ &\cdots \\ L^{(M)} &= \sigma(W^{(M)}L^{(M-1)} + b^{(M)}) \\ f(x) &= W^{(M+1)}L^{(M)} + b^{(M+1)} \end{aligned}$$

where $W^{(m)} = [W_{jk}^{(m)}] \in \mathbb{R}^{d_m \times d_{m-1}}$ and $b^{(m)} = (b_j^{(m)}) \in \mathbb{R}^{d_m}$. A predictor x_k does not impact the outcome if $W_{jk}^{(1)} = 0$ for each $j = 1, \ldots, d_1$ (predictor level sparsity - PLS). Whereas, a neuron j in layer m is inactive if $W_{jk}^{(l)} = 0$ for each $k = 1, \ldots, d_m$ and the bias $W_j^{(l)} = 0$ (unite level sparsity - ULS). Our DSS prior distribution for the parameters θ of the DNN accounts for predictor level and unite level sparsity using a latent variable representation of the form

(8)

$$W_{jk}^{(1)} = \mathcal{S}_{\lambda_1}\left(U_{jk}^{(1)}\right) \times \overbrace{\mathcal{A}_{\tau_1}\left(V_j^{(1)}\right)}^{ULS} \overbrace{\mathcal{A}_{\tau_x}\left(V_{xk}^{(1)}\right)}^{PLS} \tag{9}$$

$$b_j^{(m)} = \mathcal{S}_{\lambda_m}\left(U_j^{(m)}\right) \times \overbrace{\mathcal{A}_{\tau_m}\left(V_j^{(m)}\right)}^{ULS} \qquad \text{for} \quad m = 1, \ldots, M \tag{10}$$

$$W_{jk}^{(m)} = \mathcal{S}_{\lambda_m}\left(U_{jk}^{(m)}\right) \times \underbrace{\mathcal{A}_{\tau_m}\left(V_j^{(m)}\right)}_{ULS} \qquad \text{for} \quad m = 2, \ldots, M \text{ where} \tag{11}$$

$$\begin{aligned} U_j^{(m)}, U_{jk}^{(m)} &\overset{iid}{\sim} \pi_U \qquad \text{for } k = 1, \ldots, p_{m-1}, \\ V_j^{(m)} &\overset{iid}{\sim} \pi_V \qquad \text{for } j = 1, \ldots, p_m \text{ and } m = 1, \ldots, M+1 \\ V_{xk}^{(1)} &\overset{iid}{\sim} \pi_V \qquad \text{for } k = 1, \ldots, p_1 \end{aligned}$$

2.2 VARIATIONAL APPROXIMATION

Inference under the discrete grouped-structures DSS prior relies on the posterior distribution $\pi(\theta \mid \mathcal{D}_n)$ for θ, which, by a change- of-variables equals

$$\pi(\theta \in \mathcal{B} \mid \mathcal{D}_n) = \int_{\{U_{jk}^{(m)}, V_j^{(m)}, V_{xk}^{(1)}\}: \theta \in \mathcal{B}} d\pi(\{U_{jk}^{(m)}, V_j^{(m)}, V_{xk}^{(1)}\} \mid \mathcal{D}_n)$$

Instead of approximating the discrete grouped posterior of θ under the discrete grouped DSS prior, we approximate the posterior of $\{U_{jk}^{(m)}, V_j^{(m)}, V_{xk}^{(1)}\}$ under the continuous priors π_U, π_V and π_τ.

Surrogate Activation. For activations $A_\tau(v)$ that don't have continuous derivatives with respect to τ and v, and direct pathwise differentiation is therefor not available across all values of (τ, v), we introduce a 2nd step in our grouped DSS inference approach. We introduce, in the optimization, a surrogate function $\widetilde{A}_\tau^{(\delta)} : \mathbb{R} \to [0, +\infty)$ for the activation A_τ, which has continuous derivatives with respect to τ and v. For instance, in Example 1 we use

$$A_{\tau,\delta}^{\mathrm{S}}(v) = \begin{cases} 0, & v \le \tau, \\ \dfrac{\exp\{-\delta/(v-\tau)\}}{\exp\{-\delta/(v-\tau)\} + \exp\{-\delta/(\tau+\delta-v)\}}, & \tau < v < \tau + \delta, \\ 1, & v \ge \tau + \delta. \end{cases} \tag{12}$$

This map is continuously differentiable, bounded in $[0, 1]$. The parameter $\delta > 0$ controls the transition width. These maps alter optimization geometry but not structural semantics: throughout the paper, PIPs are computed from the hard event $V_g > \tau$. Whereas in Example 2 we approximate the ReLU, for $\delta > 0$, with

$$\widetilde{A}_\tau^{(\delta)}(v) = \frac{(v-\tau)^{1+\delta} \times I(v > \tau)}{1 + (v-\tau)^{1+\delta} \times I(v > \tau)}.$$

The smoothed activation $\widetilde{A}_\tau^{(\delta)}$ is used purely for optimization purposes for the posterior approximation over the LVR-DSS prior equation 3.

2.3 FLOW-BASED ROLE-AWARE IAF

The DSS-LVR representation transfers posterior inference from the mixed discrete–continuous BNN parameter space for θ to a continuous latent space $\xi = (U, V, \tau) \in \mathbb{R}^D$. Let $\theta(\xi)$ denote the BNN

parameters reconstructed using the chosen optimization-compatible exact-zero map. The corresponding continuous posterior is

$$\pi_A(\xi \mid \mathcal{D}_n) \propto p\{\mathcal{D}_n \mid \theta(\xi)\}\pi_U(U)\pi_V(V)\pi_\tau(\tau). \tag{13}$$

We approximate this posterior using an inverse autoregressive flow (IAF). Starting from a trainable diagonal Gaussian $z^{(0)} \sim q_0 = \mathcal{N}\{\mu_0, \mathrm{diag}(\sigma_0^2)\}$, a sequence of invertible transformations gives

$$z^{(L)} = T_L \circ \cdots \circ T_1(z^{(0)}), \qquad q_\psi(\xi) \approx \pi_A(\xi \mid \mathcal{D}_n), \tag{14}$$

with transformed density

$$\log q_\psi(\xi) = \log q_0(z^{(0)}) - \sum_{\ell=1}^{L} \log \left| \det \frac{\partial T_\ell}{\partial z^{(\ell-1)}} \right|. \tag{15}$$

To exploit the distinct roles of (U, V, τ), each transformation uses a role-aware autoregressive ordering. Let $g_\ell = (g_{\ell 1}, \ldots, g_{\ell D})$ denote the coordinate ordering in layer ℓ. We use the affine transformation

$$z_{g_{\ell j}}^{(\ell)} = z_{g_{\ell j}}^{(\ell-1)} \exp\left\{ \ell_{g_{\ell j}}^{(\ell)} \left(z_{g_{\ell,1:j-1}}^{(\ell-1)} \right) \right\} + m_{g_{\ell j}}^{(\ell)} \left(z_{g_{\ell,1:j-1}}^{(\ell-1)} \right), \tag{16}$$

where the shift and log-scale are produced jointly by a masked autoencoder (MADE), so that each coordinate depends only on preceding coordinates in the specified ordering. The resulting Jacobian is triangular, with

$$\log \left| \det \frac{\partial z^{(\ell)}}{\partial z^{(\ell-1)}} \right| = \sum_{j=1}^{D} \ell_{g_{\ell j}}^{(\ell)}. \tag{17}$$

Rather than using arbitrary coordinate permutations, successive flow layers preserve the semantic structure of DSS-LVR through the role orders (U, V, τ), (V, τ, U), and (τ, U, V). In each layer, later blocks may condition on earlier blocks, while rotating the order across layers allows each role to condition on both of the others over one autoregressive cycle.

The flow parameters ψ are learned by minimizing $\mathrm{KL}\{q_\psi(\xi) \| \pi_A(\xi \mid \mathcal{D}_n)\}$, equivalently by maximizing

$$\mathcal{L}(\psi) = E_{q_\psi(\xi)} \left[\log p\{\mathcal{D}_n \mid \theta(\xi)\} + \log \pi_U(U) + \log \pi_V(V) + \log \pi_\tau(\tau) - \log q_\psi(\xi) \right]. \tag{18}$$

After training, posterior draws are obtained by sampling $z^{(0,r)} \sim q_0$, $r = 1, \ldots, R$, and applying the fitted push-forward $\xi^{(r)} = (U^{(r)}, V^{(r)}, \tau^{(r)}) = T_{\widehat{\psi}}\{z^{(0,r)}\}$. Each draw reconstructs the corresponding DNN parameters $\theta^{(r)}$ and function $f_{\theta^{(r)}}$. Predictor inclusion probabilities are then computed from the hard structural events, $\widehat{\mathrm{PIP}}_k = R^{-1} \sum_{r=1}^{R} I\{V_{xk}^{(1,r)} > \tau_{r_x}^{(r)}\}$. Posterior DNN inference follows from the same draws, with $\widehat{f}(x) = R^{-1} \sum_{r=1}^{R} f_{\theta^{(r)}}(x)$. The same joint posterior samples also yield continuous parameter uncertainty and induced edge- and path-level posterior summaries.

3 Experiments

3.1 Experimental Setting

We evaluate whether DSS-LVR Flow can simultaneously identify relevant predictors, recover nonlinear regression functions, and induce sparse neural-network structures. Within each simulation replicate, competing methods are fitted to the same generated dataset, which is divided into 80% training and 20% test observations. The fitted MLP contains two hidden layers with 20 units each, giving the architecture $p \to 20 \to 20 \to 1$.

Our baseline data-generating mechanism uses $n = 5000$, $p = 200$, 10 active predictors, and independent covariates $X_j \sim \mathrm{Uniform}(-2.5, 2.5)$. The noiseless regression function is obtained from

$$f_{\mathrm{raw}}(x) = \sin(x_1{+}x_2){+}\sin(x_3{+}x_4){+}\sin(x_5{+}x_6){+}\sin(x_7{+}x_8){+}\sin(x_9{+}x_{10}){+}\sin(x_1{+}x_3){+}\frac{1}{2}z_{12}{+}\frac{1}{2}z_{34},$$

where

$$z_{12} = \frac{x_1 x_2}{25/12}, z_{34} = \frac{x_3 x_4}{25/12}.$$

A fixed population-level scaling is applied to f_{raw} so that the resulting signal has population standard deviation 1.5, and responses are generated as $Y = f^*(X) + \varepsilon$, with $\varepsilon \sim N(0, \sigma^2)$. The baseline setting uses $\sigma^2 = 1$.

We consider three extensions of this baseline design. First, a low-SNR setting increases the noise variance to $\sigma^2 = 4$ while retaining $n = 5000$, $p = 200$, and the same regression function. Second, a high-dimensional setting increases the ambient dimension to $p = 4000$, while retaining $n = 5000$, 10 active predictors, and $\sigma^2 = 1$. Third, to assess whether the results depend on the particular baseline nonlinear structure, we use a paired-product trigonometric design with

$$f_{\text{raw}}^{\text{PP}}(x) = \sin(x_1 x_2) + \cos(x_3 x_4) + \sin^2(x_5 x_6) + \cos^2(x_7 x_8) + \sin(x_9 x_{10}) + x_1 x_2 + x_3 x_4.$$

For this design, the full noiseless signal is centered and rescaled within each generated dataset to standard deviation 1.5, following the original paired-product simulator. All multilayer settings use 20 independently generated datasets.

For the baseline comparison, we compare DSS-LVR with six alternative sparse BNN approaches: LBBNN-LRT, LBBNN-FLOW, ISLaB-FLOW, SS-GL, SS-GHS, and IS-ANN-L1. The low-SNR, high-dimensional, and paired-product experiments are used to characterize the behavior of DSS-LVR across changes in signal strength, ambient dimensionality, and nonlinear structure. Additional details on the simulation designs are provided in Appendix B.1, while method-specific specifications and optimization settings are given in Appendices B.2 and B.3.

Metrics. We evaluate variable-selection accuracy, function recovery, structural sparsity, and computational cost.

Variable-selection performance is summarized by the true-positive rate, false-positive rate, and classification accuracy, defined as $\text{TPR} = \text{TP}/(\text{TP} + \text{FN})$, $\text{FPR} = \text{FP}/(\text{FP} + \text{TN})$, and $\text{Accuracy} = (\text{TP} + \text{TN})/p$, respectively.

Function recovery is assessed on held-out observations against the true noiseless regression function f^*. We report $\text{MSE} = n_{\text{test}}^{-1} \sum_{i \in \mathcal{D}_{\text{test}}} \{\hat{f}(x_i) - f^*(x_i)\}^2$ and $R^2 = 1 - \sum_i \{\hat{f}(x_i) - f^*(x_i)\}^2 / \sum_i \{f^*(x_i) - \bar{f}^*\}^2$. Lower MSE and higher R^2 indicate better recovery of the underlying regression function.

For multilayer networks, structural sparsity is evaluated at the parameter, edge, and path levels. We define $D_{\text{param}} = N_{\text{retained parameters}}/N_{\text{dense parameters}}$, $D_{\text{edge}} = N_{\text{active edges}}/N_{\text{dense edges}}$, and $D_\pi = N_{\text{active input-output paths}}/N_{\text{dense input-output paths}}$.

3.2 Results

Predictor selection with structural sparsity. The primary advantage of DSS-LVR is that predictor selection and network compression occur simultaneously rather than as separate post-processing objectives. Under the baseline nonlinear design, DSS-LVR recovered most active predictors while excluding the large majority of inactive predictors, with TPR $= 0.935$, FPR $= 0.043$, and selection accuracy $= 0.956$. This predictor-level discrimination was accompanied by substantial compression of the fitted network: only 3.68% of the dense parameterization and 3.80% of the possible edges were retained, while the proportion of active input–output paths was reduced to 0.54%. Thus, accurate predictor recovery did not require maintaining a dense internal representation.

The competing methods exhibited two qualitatively different failure modes. LBBNN-LRT, LBBNN-FLOW, ISLaB-FLOW, and IS-ANN-L1 retained essentially all predictors, producing TPRs near one but FPRs also near one. Their fitted networks therefore remained substantially denser than DSS-LVR. In contrast, SS-GL and SS-GHS produced nearly empty effective networks, but this numerical sparsity was accompanied by essentially no recovery of the nonlinear signal. DSS-LVR therefore occupied a distinct regime in which predictor discrimination and structural sparsification were achieved jointly rather than by either retaining nearly the full predictor set or collapsing the fitted network.

Table 1: Comparison under the baseline sine-interaction simulation. Entries report the mean with 95% confidence interval across 20 simulation replicates.

Method	Runs	MSE	R^2	TPR	FPR	Acc.	D_{param}	D_{edge}	D_π	Time (min)
DSS-LVR	20/20	0.993 [0.828,1.158]	0.560 [0.488,0.631]	0.935 [0.908,0.962]	0.043 [-0.013,0.099]	0.956 [0.903,1.009]	0.0368 [0.0222,0.0513]	0.0380 [0.0224,0.0536]	0.0054 [0.0029,0.0080]	5.32
LBBNN-LRT	20/20	3.444 [3.356,3.533]	-0.530 [-0.574,-0.487]	1.000 [1.000,1.000]	1.000 [1.000,1.000]	0.050 [0.050,0.050]	0.3754 [0.3717,0.3791]	0.3836 [0.3803,0.3869]	0.0865 [0.0777,0.0953]	41.01
LBBNN-FLOW	20/20	2.987 [2.867,3.108]	-0.327 [-0.384,-0.271]	1.000 [1.000,1.000]	0.999 [0.999,1.000]	0.051 [0.050,0.051]	0.3470 [0.3418,0.3522]	0.3555 [0.3508,0.3602]	0.0735 [0.0660,0.0810]	186.59
ISLaB-FLOW	20/20	3.316 [3.200,3.432]	-0.472 [-0.520,-0.425]	1.000 [1.000,1.000]	1.000 [1.000,1.000]	0.050 [0.050,0.050]	0.2739 [0.2671,0.2807]	0.3065 [0.3028,0.3102]	0.0911 [0.0854,0.0969]	191.08
SS-GL	20/20	2.255 [2.216,2.294]	-0.001 [-0.002,-0.000]	–	–	–	0.0000 [0.0000,0.0000]	0.0000 [0.0000,0.0000]	0.0000 [0.0000,0.0000]	2.64
SS-GHS	20/20	2.255 [2.216,2.295]	-0.001 [-0.002,-0.000]	–	–	–	0.0000 [0.0000,0.0000]	0.0000 [0.0000,0.0000]	0.0000 [0.0000,0.0000]	3.89
IS-ANN-L1	20/20	2.327 [2.227,2.426]	-0.033 [-0.078,0.011]	1.000 [1.000,1.000]	1.000 [1.000,1.000]	0.050 [0.050,0.050]	0.1525 [0.1391,0.1658]	0.1525 [0.1391,0.1658]	0.0155 [0.0141,0.0168]	9.06

Table 2: Sensitivity analysis for DSS-LVR across nonlinear structure, signal-to-noise ratio, and dimensionality. Entries report the mean with 95% confidence interval across 20 simulation replicates.

Setting	Runs	MSE	R^2	TPR	FPR	Acc.	D_{param}	D_{edge}	D_π
Baseline sine-interaction	20/20	0.993 [0.828,1.158]	0.560 [0.488,0.631]	0.935 [0.908,0.962]	0.043 [-0.013,0.099]	0.956 [0.903,1.009]	0.0368 [0.0222,0.0513]	0.0380 [0.0224,0.0536]	0.0054 [0.0029,0.0080]
Paired-product	20/20	0.238 [0.229,0.248]	0.895 [0.891,0.900]	0.400 [0.400,0.400]	0.000 [0.000,0.000]	0.970 [0.970,0.970]	0.0184 [0.0079,0.0289]	0.0189 [0.0084,0.0294]	0.0023 [0.0003,0.0043]
Low SNR	20/20	2.240 [2.186,2.294]	0.006 [-0.008,0.021]	0.190 [0.088,0.292]	0.198 [0.101,0.295]	0.771 [0.683,0.860]	0.0001 [-0.0001,0.0003]	0.0031 [0.0025,0.0037]	0.000030 [0.000007,0.000053]
High-dimensional	20/20	2.232 [2.195,2.270]	-0.001 [-0.002,-0.000]	0.300 [0.115,0.485]	0.276 [0.095,0.457]	0.723 [0.543,0.903]	0.0000 [0.0000,0.0000]	0.0009 [0.0004,0.0014]	0.000001 [0.000000,0.000003]

The same structural behavior persisted under the paired-product design. DSS-LVR selected no inactive predictors at the median-probability threshold and retained only 1.84% of the dense parameterization, 1.89% of the possible edges, and 0.23% of the possible input–output paths. These results indicate that the group-structured posterior acts not only as a predictor-selection mechanism but also as an effective mechanism for inducing sparse computation throughout a multilayer network.

Nonlinear function recovery under sparse representations. Strong structural compression did not necessarily prevent recovery of the underlying nonlinear regression function. Under the baseline design, DSS-LVR obtained an MSE of 0.993 and $R^2 = 0.560$, substantially outperforming the competing methods in function recovery while retaining a much smaller effective network. Nevertheless, function recovery was less complete than predictor recovery in this setting: the active predictor set was recovered with high accuracy even though approximately half of the variation in the noiseless regression function remained unexplained. This distinction shows that support recovery and functional recovery are related but not equivalent objectives in sparse neural networks.

The paired-product design provides the complementary case. DSS-LVR achieved MSE $= 0.238$ and $R^2 = 0.895$ while its predictor-level TPR at the median-probability threshold was only 0.400, with FPR $= 0$. Hence, substantially better reconstruction of the regression surface was obtained even though only a subset of the nominally active predictors crossed the posterior inclusion threshold. Taken together, the two nonlinear designs demonstrate that predictor recovery and function recovery need not move in parallel: a model may identify the active support accurately without completely reconstructing the nonlinear function, or recover the function accurately using a particularly sparse subset of the available predictive structure.

When the noise variance was increased, DSS-LVR produced an almost empty effective parameterization, with $D_{\mathrm{param}} = 0.0001$, while TPR decreased to 0.190 and R^2 to 0.006. A similar collapse of the effective network occurred in the high-dimensional setting. With $p = 4000$, parameter density was effectively zero and edge density was 0.0009, while function recovery disappeared ($R^2 = -0.001$). Predictor-level discrimination also deteriorated, with TPR $= 0.300$ and FPR $= 0.276$. These results indicate that under less informative regimes, strong structural compression can be accompanied by substantial losses in both support and function recovery.

Posterior fidelity in the shallow network. The shallow-network experiment evaluates the posterior fidelity of DSS-LVR Flow against a matched MCMC reference. The reference posterior was

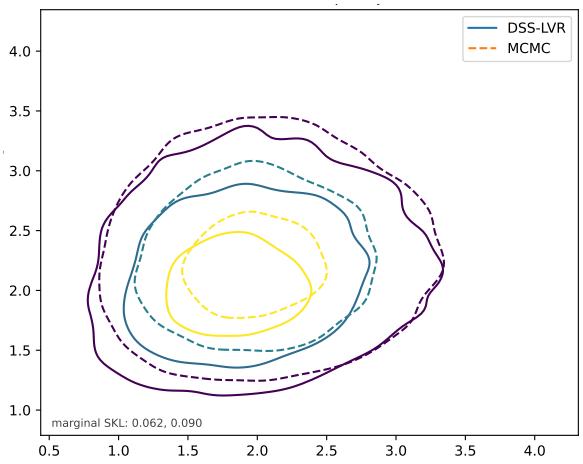


Figure 2: Joint posterior comparison between DSS-LVR Flow and MCMC

obtained from eight independent Metropolis–Hastings chains of 50,000 iterations each, with the first 10,000 iterations of each chain discarded as warmup; further sampling details are provided in Appendix B.3. Across the 20 matched simulation replicates, the Active SKL for true-active slab components averaged 0.203 (median 0.200), while the Zero JS (Lin, 1991) for true-inactive components averaged 0.0298; definitions of these posterior-recovery metrics are provided in Appendix B.5. The close agreement indicates that DSS-LVR Flow preserved both the continuous posterior variation among active effects and the structural uncertainty associated with excluded effects.

The joint posterior comparison provides a complementary view of dependence recovery. For the selected true-active pair shown in Figure 2, DSS-LVR closely reproduced the location, scale, and dependence pattern obtained under MCMC, with joint SKL $= 0.167$ and marginal SKLs of 0.062 and 0.090. The aggregate divergences across simulation replicates, rather than this selected visualization, provide the primary quantitative evidence for posterior fidelity. Together, these results show that DSS-LVR Flow retains the main continuous and structural features of the DSS posterior in the setting where an external MCMC reference is available.

4 Conclusion

We introduced DSS-LVR Flow, a latent-variable representation of group-structured discrete spike-and-slab priors for sparse Bayesian neural networks. The framework uses optimization-compatible exact-zero mappings to support gradient-based posterior learning while preserving exact structural inclusion semantics. It further incorporates a role-aware inverse autoregressive flow to construct a joint posterior approximation over continuous parameters, structural latent variables, and their shared thresholds. In the baseline multilayer experiments, DSS-LVR Flow simultaneously achieved selective predictor recovery and substantial network compression, accurately identifying most active predictors while retaining only a small fraction of parameters, edges, and complete input–output paths. Compared with the sparse BNN methods considered in our experiments, DSS-LVR Flow provided a stronger overall balance among predictor selection, nonlinear function recovery, and structural sparsity.

Limitations. The current results also identify settings in which the proposed sparsity mechanism remains fragile. Under reduced signal-to-noise ratio and high ambient dimensionality, DSS-LVR can shrink the effective network too aggressively, with corresponding deterioration in both predictor and function recovery. Improving the calibration of structural shrinkage in weak-information regimes is therefore an important direction for further work.

REFERENCES

Eirik Høyheim, Lars Skaaret-Lund, Solve Sæbø, and Aliaksandr Hubin. Explainable bayesian deep learning through input-skip latent binary bayesian neural networks. *Journal of Artificial Intelligence Research*, 86, 2026. doi: 10.1613/jair.1.20433.

Hemant Ishwaran and J. Sunil Rao. Spike and Slab Variable Selection: Frequentist and Bayesian Strategies. *The Annals of Statistics*, 33(2):730–773, 2005. doi: 10.1214/009053604000001147.

Sanket Jantre, Shrijita Bhattacharya, and Tapabrata Maiti. Spike-and-slab shrinkage priors for structurally sparse bayesian neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 36(6):11176–11188, 2025. doi: 10.1109/TNNLS.2024.3485529.

Diederik P. Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved Variational Inference with Inverse Autoregressive Flow. In *Advances in Neural Information Processing Systems*, volume 29, pp. 4743–4751, 2016.

J. Lin. Divergence measures based on the Shannon entropy. *IEEE Transactions on Information Theory*, 37(1):145–151, January 1991. ISSN 1557-9654. doi: 10.1109/18.61115. URL `https://ieeexplore.ieee.org/document/61115`.

Akanksha Mishra, Wei Xia, and Clint Pazhayidam George. A weight-sharing bayesian neural network for consistent feature selection with applications in cancer gene expression data. *BMC Bioinformatics*, 27:79, 2026. doi: 10.1186/s12859-026-06397-0.

Lars Skaaret-Lund, Geir Storvik, and Aliaksandr Hubin. Sparsifying bayesian neural networks with latent binary variables and normalizing flows. *Transactions on Machine Learning Research*, 2024. URL `https://openreview.net/forum?id=d6kqUKzG3V`.

A Additional Methodological Details

A.1 Group-Structured DSS-LVR Construction

A.2 Predictor and Hidden-Unit Groups

A.3 Induced Edge and Path Structures

A.4 Exact-Zero Relaxation and Hard Posterior Summaries

A.5 Role-Aware IAF Construction

A.6 Multilayer Extension and Structural Invariance

B Complete Experimental Protocol

B.1 Simulation Design and Data-Generating Mechanisms

B.2 Compared Methods

B.3 Optimization and Posterior Sampling

For the shallow-network posterior-fidelity experiment, DSS-LVR Flow was compared with a Metropolis–Hastings MCMC reference fitted to the same simulated datasets. For each replicate, we ran eight independent MCMC chains, each for 50,000 iterations, and discarded the first 10,000 iterations of each chain as warmup. The retained draws were used to construct the MCMC reference posterior for the continuous slab and structural inclusion quantities. MCMC was used only for the shallow-network validation, where the posterior dimension remains sufficiently controlled for direct posterior comparison; it was not used as a reference for the multilayer experiments.

B.4 Evaluation Metrics and Confidence Intervals

For each regime–method combination, summary statistics are computed over successful fits with finite required metrics. If $x_1, \ldots, x_m$ are the available replicate-level values, we report the sample mean and the two-sided 95% Student-t interval $\bar{x} \pm t_{0.975,m-1} s/\sqrt{m}$. The intervals are not truncated to the natural range of bounded metrics, so confidence limits for quantities such as TPR, FPR, or accuracy may extend slightly below zero or above one. Failed fits and fits with non-finite required metrics are excluded, while finite extreme values are retained.

B.5 Evaluation Metrics and Confidence Intervals

For the shallow-network MCMC validation, posterior fidelity is summarized by Active SKL and Zero JS. For a true-active component g, let p_g and q_g denote the DSS-LVR Flow and MCMC conditional slab-strength densities, respectively, given that the component is active. Their symmetric Kullback–Leibler divergence is

$$D_{\mathrm{SKL}}(p_g, q_g) = \frac{1}{2}\,\mathrm{KL}(p_g \| q_g) + \frac{1}{2}\,\mathrm{KL}(q_g \| p_g).$$

Active SKL summarizes this divergence across the true-active components, with smaller values indicating closer agreement between the DSS-LVR Flow and MCMC continuous posteriors.

For true-inactive components, posterior structural agreement is measured using the Jensen–Shannon divergence between the Bernoulli inclusion distributions implied by the DSS-LVR Flow and MCMC posterior inclusion probabilities. Let P_g and Q_g denote these two Bernoulli distributions and $M_g = (P_g + Q_g)/2$. Then

$$D_{\mathrm{JS}}(P_g, Q_g) = \frac{1}{2}\,\mathrm{KL}(P_g \| M_g) + \frac{1}{2}\,\mathrm{KL}(Q_g \| M_g).$$

Zero JS summarizes this discrepancy across true-inactive components. Lower values therefore indicate closer agreement in posterior exclusion probabilities. The joint posterior visualization ad-

ditionally reports a two-dimensional symmetric KL divergence computed for the selected pair of true-active slab components.

For each regime–method combination, summary statistics are computed over successful fits with finite required metrics. If $x_1, \ldots, x_m$ are the available replicate-level values, we report the sample mean and the two-sided 95% Student-t interval

$$\bar{x} \pm t_{0.975,m-1} \frac{s}{\sqrt{m}}.$$

The intervals are not truncated to the natural range of bounded metrics, so confidence limits for quantities such as TPR, FPR, or accuracy may extend slightly below zero or above one. Failed fits and fits with non-finite required metrics are excluded, while finite extreme values are retained.

B.6 Computing Environment and Runtime Protocol

Formal experiments were submitted as Slurm arrays on the Minnesota Supercomputing Institute (MSI) cluster. The baseline and DSS-LVR extension jobs used seeds 561–580. DSS-LVR, SS-GL, and SS-GHS jobs were submitted to the `msigpu` partition with one requested GPU, two CPU cores per task, and 12 GB of memory, with a 6-hour time limit; the high-dimensional DSS-LVR jobs used 24 GB and a 12-hour limit. LBBNN-LRT, LBBNN-FLOW, ISLaB-FLOW, and IS-ANN-L1 were submitted to the `msismall` partition with one CPU core, 8 GB of memory, and an 8-hour time limit. OpenMP, MKL, and OpenBLAS thread counts were restricted to the allocated CPU count. The runtime reported in Table 1 is each method's recorded fitting time (`runtime_sec`) converted to minutes and averaged across successful fits; it does not include queue waiting time.

C Additional Experimental Results

C.1 Additional Sparsity Regimes

C.2 Role-Aware IAF Ordering and Depth Ablations

C.3 Posterior-Parameterization Ablations

C.4 Baseline Hyperparameter Sensitivity

C.5 Additional Structural-Sparsity Results

C.6 Runtime and Scaling Results

D Implementation and Reproducibility Details

D.1 Model and Flow Implementation

D.2 Random Seeds and Replicate Construction

The formal simulation experiments use data seeds 561–580, giving 20 independently generated datasets per reported regime. All methods in the baseline comparison are fitted to the same dataset for a given data seed. DSS-LVR uses deterministic fit-seed offsets from the data seed, while the external SS and R-based baselines use separate deterministic seed offsets, so stochastic optimization is reproducible without changing the generated data.

D.3 Software and Hardware

The archived reproducibility environment uses the Conda environment `nf311` with Python 3.11.15, NumPy 2.0.2, pandas 2.2.3, SciPy 1.14.1, scikit-learn 1.5.2, and PyTorch 2.11.0+cpu. The R baselines use R 4.4.2 through the MSI module `r-rich/4.4.2_msi1.2`, with `LBBNN` 0.1.6 and R `torch` 0.17.0; the recorded R session used FlexiBLAS/OpenBLAS-OpenMP. Exact processor and accelerator models were not fixed in the archived submission scripts. The archived Python environment is a CPU-only PyTorch build, so `device=auto` resolves to CPU in this environment even when a Slurm job requests a GPU resource.

D.4 CODE ORGANIZATION

The formal experiment configuration is defined in `final_mlp_config.py`. Per-replicate jobs are dispatched by `run_final_mlp_task.py`; DSS-LVR fitting and evaluation are handled by `dss_runner.py`; and replicate-level outputs are summarized by `aggregate_final_mlp.py`. Slurm submission and worker scripts are stored under `hpc/`. The archived environment additionally contains the Conda specification, Python and R package inventories, and session information used for the formal run.