Journal of Machine Learning Research 20 (2019) 1-46 Submitted 3/19; Revised 10/19; Published 10/19

# Model Selection in Bayesian Neural Networks via Horseshoe Priors

**Soumya Ghosh** GHOSHSO@US.IBM.COM
*MIT-IBM Watson AI Lab and Center for Computational Health, IBM Research, Cambridge, MA 02142.*

**Jiayu Yao** JIY328@G.HARVARD.EDU
**Finale Doshi-Velez** FINALE@SEAS.HARVARD.EDU
*School of Engineering and Applied Sciences, Harvard University, Cambridge, MA 02138.*

**Editor:** Mohammad Emtiyaz Khan

## Abstract

The promise of augmenting accurate predictions provided by modern neural networks with well-calibrated predictive uncertainties has reinvigorated interest in Bayesian neural networks. However, model selection—even choosing the number of nodes—remains an open question. Poor choices can severely affect the quality of the produced uncertainties. In this paper, we explore continuous shrinkage priors, the horseshoe, and the regularized horseshoe distributions, for model selection in Bayesian neural networks. When placed over node pre-activations and coupled with appropriate variational approximations, we find that the strong shrinkage provided by the horseshoe is effective at turning off nodes that do not help explain the data. We demonstrate that our approach finds compact network structures even when the number of nodes required is grossly over-estimated. Moreover, the model selection over the number of nodes does not come at the expense of predictive or computational performance; in fact, we learn smaller networks with comparable predictive performance to current approaches. These effects are particularly apparent in sample-limited settings, such as small data sets and reinforcement learning.

**Keywords:** Bayesian Neural Networks, Model Selection, Horseshoe Priors, Variational Inference, Structured approximations

## 1. Introduction

Bayesian neural networks (BNNs) treat the weights in a neural network as random variables. By performing posterior inference on these weights, BNNs can avoid overfitting in the regime of small data, provide posterior uncertainty estimates, and model a large class of stochastic functions with heteroskedastic and multi-modal noise. These properties have resulted in BNNs being adopted in applications ranging from Bayesian optimization (Springenberg et al., 2016) and active learning (Gal et al., 2016a) to reinforcement learning (Blundell et al., 2015; Depeweg et al., 2017).

While there have been many recent advances in training BNNs (Hernández-Lobato and Adams, 2015; Blundell et al., 2015; Gal and Ghahramani, 2016; Louizos and Welling, 2016; Hernandez-Lobato et al., 2016), model-selection has received relatively less attention. Unfortunately, the consequences for a poor choice of architecture are severe: too few nodes,

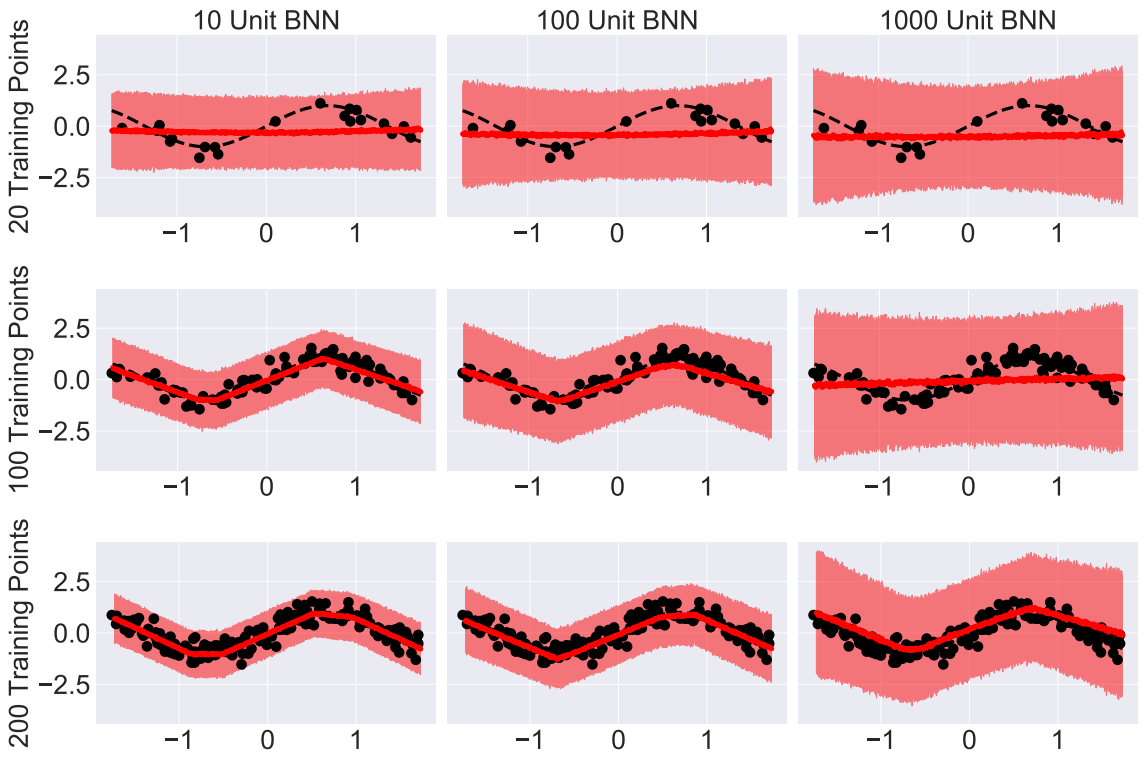


Figure 1: **Predictive distributions** from a single layer BNN with $\mathcal{N}(0, 1)$ priors over weights, ReLU activations, containing ten, hundred, and thousand units, trained on noisy samples (in black) from a smooth one dimensional function shown in black, $y = \mathcal{N}(sin(x), 0.1)$. With fixed data increasing BNN capacity leads to large predictive uncertainties.

and the BNN will not be flexible enough to model the function of interest; too many nodes, and the BNN predictions will have large variance. We note that these Bayesian model selection concerns are subtly different from overfitting and underfitting concerns that arise from maximum likelihood training: here, more expressive models (e.g. those with more nodes) require more data to concentrate the posterior. When there is insufficient data, the posterior uncertainty over the BNN weights will remain large, resulting in large variances in the BNN's predictions. We illustrate this issue in Figure 1, where we see a BNN trained with a large number parameters has higher variance around its predictions than one with fewer. Thus, the core concern of Bayesian model selection is to identify a model class expressive enough that it can explain the observed data set, but not so expressive that it can explain everything (Rasmussen and Ghahramani, 2001; Murray and Ghahramani, 2005).

Model selection in BNNs is challenging because the number of nodes in a layer is a discrete quantity, forcing practitioners to perform onerous searches over different layer sizes. In this work, we demonstrate that we can perform computationally-efficient and statistically-

effective model selection in Bayesian neural networks by placing horseshoe (Carvalho et al., 2009) and related priors over the variance of weights incident to each node in the network. These priors can be interpreted as a continuous relaxation of a spike-and-slab approach that would assign a discrete on-off variable to each node. The prior has heavy tails and supports both zero values and large values. Thus, if we fix the mean of the incident weights to be zero, nodes with small variance parameters are effectively turned off—all incident weights will be close to zero—while nodes with large variance parameters can be interpreted as active. In this way, we can perform model selection over the number of nodes required in a Bayesian neural network. The continuous relaxation provided by these priors keep the model differentiable; with appropriate parameterization, we can take advantage of recent advances in variational inference (e.g., Kingma and Welling, 2014) for training. Especially in sample-limited settings, we demonstrate that our approach returns compact network structures even when the required number of nodes in the network is grossly over-estimated; these compact structures sacrifice little—and sometimes even improve—predictive performance. We demonstrate the value of these prediction gains through a reinforcement learning application.

Finally, we note that aspects of this work have appeared in (Ghosh and Doshi-Velez, 2017a,b; Ghosh et al., 2018); here we organize these results and provide a cohesive picture on priors for model selection in Bayesian neural networks.

## 2. Bayesian Neural Networks

A deep neural network with $L-1$ hidden layers is parameterized by a set of weight matrices $\mathcal{W} = \{W_l\}_1^L$, with each weight matrix $W_l$ being of size $\mathbb{R}^{(K_{l-1}+1)\times K_l}$ where $K_l$ is the number of units (excluding the bias) in layer $l$. The neural network maps an input $x \in \mathbb{R}^{D\times 1}$ to a response $f(\mathcal{W}, x)$ by recursively applying the transformation $h(W_l^T[z_{l-1}; 1])$, where the vector $z_l$ is the input with the initial input $z_0 = x$, and $h$ is a point-wise non-linearity, for instance the rectified-linear function, $h(a) = \max(0, a)$.

A Bayesian neural network captures uncertainty in the weight parameters $\mathcal{W}$ by endowing them with distributions $\mathcal{W} \sim p(\mathcal{W})$. Given a data set of $N$ observation-response pairs $\mathcal{D} = \{x_n, y_n\}_{n=1}^N$, we are interested in estimating the posterior distribution,

$$p(\mathcal{W} \mid \mathcal{D}) = \frac{p(\mathcal{W}) \prod_{n=1}^N p(y_n \mid f(\mathcal{W}, x_n))}{p(\mathcal{D})},$$

and leveraging the learned posterior for predicting responses to unseen data $x_*$, $p(y_* \mid x_*) = \int p(y_* \mid f(\mathcal{W}, x_*))p(\mathcal{W} \mid \mathcal{D})d\mathcal{W}$. The prior $p(\mathcal{W})$ ostensibly allows one to encode problem-specific beliefs as well as general properties about weights. Specifying meaningful priors for Bayesian neural networks, however, remains challenging.

### 2.1. Priors over neural network weights

Zero-mean Gaussian distributions are perhaps the most commonly used prior on weights. They have been explored both in the classic works of (MacKay, 1992; Neal, 1993, 1997) as well as in more recent approaches (Louizos and Welling, 2016; Hernández-Lobato and Adams, 2015; Louizos and Welling, 2017; Pawlowski et al., 2017). Log uniform (Kingma

et al., 2015), two component scale mixture of Gaussians (Blundell et al., 2015), layer wise structured Gaussians (Louizos and Welling, 2016; Sun et al., 2017), as well as data space priors (Hafner et al., 2019) have been proposed. Buntine and Weigend (1991) explored entropic priors on BNN weights. While zero mean fixed variance Gaussian priors on weights are a convenient choice they can lead to undesirable pathologies (Figures 1, 4) — inflated predictive uncertainties, especially when used in conjunction with fully factorized variational inference. To see why such pathologies arise, note that the prior function, induced by a network with standard Gaussian weight priors, when evaluated at two data points exhibits a covariance that scales with the number of units in the penultimate layer (Williams, 1997). Further, since neural networks are over-parameterized, the prior can overwhelm the data, especially in the small data, wide network limit leading to high predictive uncertainties in these regimes. A work around is to scale the prior variances by the number of inputs as prescribed by Neal (1997). Alternatively, one could exploit the fact that weights of a network are effectively lower dimensional and attempt to parametrize the network in this lower dimensional space (Pradier et al., 2018; Izmailov et al., 2019). Such a parameterization may prevent the prior from overwhelming small amounts of data. Another option for avoiding such issues is to use hierarchical priors over weights (MacKay, 1995). This is the direction explored in this paper. In particular, we employ the horseshoe distribution (Carvalho et al., 2009), an example of a global-local shrinkage prior (Polson and Scott, 2010), that in addition to alleviating the inflated predictive uncertainty issue allows us to recover smaller networks.

## 3. Bayesian Neural Networks with Structured Sparsity

Let the node weight vector $w_{kl} \in \mathbb{R}^{(K_{l-1}+1)}$ denote the set of weights incident into unit $k$ of hidden layer $l$. There exist several ways to induce group sparsity, that is, turn all the weights incident to a node small. In this work, we focus on the use of horseshoe priors to induce group sparsity.

### 3.1. Horseshoe Priors

We assume that each node weight vector $w_{kl}$ is conditionally independent and distributed according to a Gaussian scale mixture,

$$w_{kl} \mid \tau_{kl}, \upsilon_l \sim \mathcal{N}(0, (\tau_{kl}^2 \upsilon_l^2)\mathbb{I}), \quad \tau_{kl} \sim C^+(0, b_0), \quad \upsilon_l \sim C^+(0, b_g). \tag{1}$$

Here, $\mathbb{I}$ is an identity matrix, $a \sim C^+(0, b)$ is the Half-Cauchy distribution with density $p(a|b) = 2/\pi b(1 + (a^2/b^2))$ for $a > 0$, $\tau_{kl}$ is a unit specific scale parameter, while the scale parameter $\upsilon_l$ is shared across the layer.

The distribution over weights in Equation 1 defines the horseshoe prior (Carvalho et al., 2009). It exhibits Cauchy-like flat, heavy tails while maintaining an infinitely tall spike at zero. Consequently, it has the desirable property of allowing sufficiently large node weight vectors $w_{kl}$ to escape un-shrunk—by having a large scale parameter—while providing severe shrinkage to smaller weights. This is in contrast to Lasso style regularizers and their Bayesian counterparts that provide uniform shrinkage to all weights. By forcing the weights incident on a unit to share scale parameters, the prior in Equation 1 induces sparsity at the unit level, turning off units that are unnecessary for explaining the data well. Intuitively,

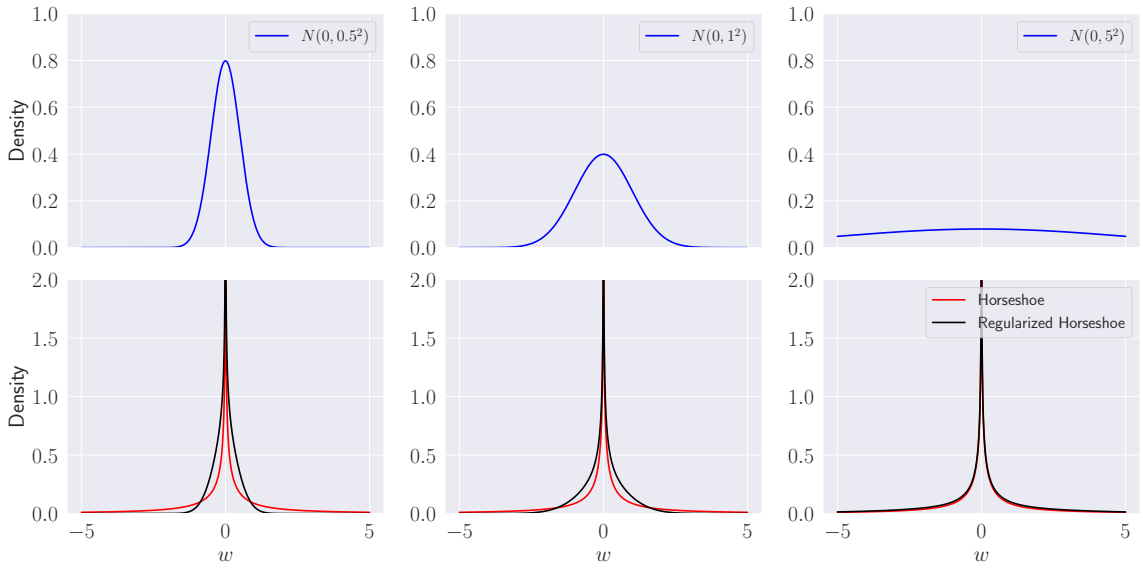


Figure 2: **Horseshoe and regularized horseshoe densities**. Both densities exhibit a spike at zero. While the horseshoe exhibits heavy Cauchy-like tails, the tails of the regularized Horseshoe are (soft) truncated. The soft truncation is governed by a zero-mean Gaussian density $\mathcal{N}(0, c^2)$, shown in the top row. Smaller values of $c$ lead to sharper truncation. To model uncertainty in our beliefs about the truncation strength $c$, we place an inverse Gamma prior, $c^2 \sim \text{Inv-Gamma}(c_a, c_b)$. In the above plots, the horseshoe parameters $b_0$ and $b_g$ are both set to one.

the shared layer wide scale $\upsilon_l$ pulls all units in layer $l$ to zero, while the heavy tailed unit specific $\tau_{kl}$ scales allow some of the units to escape the shrinkage.

### 3.2. Regularized Horseshoe Priors

While the horseshoe prior has some good properties, when the amount of training data is limited, units with essentially no shrinkage can produce large weights and adversely affect generalization performance of HS-BNNs, with minor perturbations of the data leading to vastly different predictions. To deal with this issue, we consider a related alternative, the regularized horseshoe prior (Piironen and Vehtari, 2017). Under this prior $w_{kl}$ is drawn from,

$$w_{kl} \mid \tau_{kl}, \upsilon_l, c \sim \mathcal{N}(0, (\tilde{\tau}_{kl}^2 \upsilon_l^2)\mathbb{I}), \quad \tilde{\tau}_{kl}^2 = \frac{c^2 \tau_{kl}^2}{c^2 + \tau_{kl}^2 \upsilon_l^2}. \tag{2}$$

Note that for the weight node vectors that are strongly shrunk to zero, we will have tiny $\tau_{kl}^2 \upsilon_l^2$. When, $\tau_{kl}^2 \upsilon_l^2 \ll c^2$, $\tilde{\tau}_{kl}^2 \to \tau_{kl}^2 \upsilon_l^2$, recovering the original horseshoe prior. On the other hand, for the un-shrunk weights $\tau_{kl}^2 \upsilon_l^2$ will be large, and when $\tau_{kl}^2 \upsilon_l^2 \gg c^2$, $\tilde{\tau}_{kl}^2 \to c^2$. Thus, these weights under the regularized horseshoe prior follow $w_{kl} \sim \mathcal{N}(0, c^2 \mathbb{I})$ and $c$ acts as a weight decay hyper-parameter. We place a Inv-Gamma$(c_a, c_b)$ prior on $c^2$. Another way to

interpret the regularized horseshoe prior is to observe that,

$$w_{kl} \mid \tau_{kl}, \upsilon_l, c \propto \mathcal{N}(0, (\tau_{kl}^2 \upsilon_l^2)\mathbb{I})\mathcal{N}(0, c^2\mathbb{I}).$$

This view makes it clear that the regularized variant soft thresholds the horseshoe tails by penalizing large values of $w_{kl}$ through $\mathcal{N}(0, c^2\mathbb{I})$. See Figure 2 for a visual comparison of the densities of the horseshoe and the regularized horseshoe distributions for different values of $c$ and Figure 24 for a comparison with the Laplace distribution. In the experimental section, we find that the regularizedHS-BNN does indeed improve generalization over HS-BNN.

### 3.3. Half-Cauchy re-parameterization for variational learning.

Both the standard and regularized horseshoe priors involve the Half-Cauchy distribution. While a direct parameterization of the Half-Cauchy distribution in Equation 1 is possible, it leads to challenges during variational learning: standard exponential family variational approximations struggle to capture the thick Cauchy tails, and using Cauchy approximating family leads to high variance gradients.

Therefore, we use a more convenient auxiliary variable parameterization (Wand et al., 2011),

$$a \sim C^+(0, b) \iff a^2 \mid \lambda \sim \text{Inv-Gamma}(\frac{1}{2}, \frac{1}{\lambda}); \lambda \sim \text{Inv-Gamma}(\frac{1}{2}, \frac{1}{b^2}),$$

where $v \sim \text{Inv-Gamma}(a, b)$ is the Inverse Gamma distribution with density $p(v) \propto v^{-a-1}\exp\{-b/v\}$ for $v > 0$. Since the number of output units is fixed by the problem at hand, there is no need for sparsity inducing prior for the output layer. We place isotropic Gaussian priors, $w_{kL} \sim \mathcal{N}(0, \kappa^2\mathbb{I})$ with vague hyper-priors $\kappa \sim C^+(0, b_\kappa = 5)$ on the output layer weights.

The joint distribution of the horseshoe Bayesian neural network is then given by,

$$\begin{aligned} p_{\text{HS}}(\mathcal{D}, \theta) =& r(\kappa, \rho_\kappa \mid b_\kappa) \prod_{k=1}^{K_L} \mathcal{N}(w_{kL} \mid 0, \kappa^2\mathbb{I}) \prod_{l=1}^{L} \Big\{ r(\upsilon_l, \vartheta_l \mid b_g) \\ & \prod_{k=1}^{K_l} r(\tau_{kl}, \lambda_{kl} \mid b_0)\mathcal{N}(w_{kl} \mid 0, (\tau_{kl}^2\upsilon_l^2)\mathbb{I}) \Big\} \prod_{n=1}^{N} p(y_n \mid f(\mathcal{W}, x_n)), \end{aligned}$$

where $p(y_n|f(\mathcal{W}, x_n))$ is an appropriate likelihood function, and,

$$r(a, \lambda|b) = \text{Inv-Gamma}(a^2|1/2, 1/\lambda)\text{Inv-Gamma}(\lambda|1/2, 1/b^2),$$

with $\theta = \{\mathcal{W}, \mathcal{T}, \kappa^2, \rho_\kappa\}$, $\mathcal{T} = \{\{\tau_{kl}\}_{k=1,l=1}^{K,L}, \{\upsilon_l\}_{l=1}^{L}, \{\lambda_{kl}\}_{k=1,l=1}^{K,L}, \{\vartheta_l\}_{l=1}^{L}\}$. An analogous reparameterization can be applied to the Bayesian neural network with regularized horseshoe priors:

$$\begin{aligned} p_{\text{regHS}}(\mathcal{D}, \theta) =& p(c \mid c_a, c_b) r(\kappa, \rho_\kappa \mid b_\kappa) \prod_{k=1}^{K_L} \mathcal{N}(w_{kL} \mid 0, \kappa^2\mathbb{I}) \prod_{l=1}^{L} \Big\{ r(\upsilon_l, \vartheta_l \mid b_g) \\ & \prod_{k=1}^{K_l} r(\tau_{kl}, \lambda_{kl} \mid b_0)\mathcal{N}(w_{kl} \mid 0, (\tilde{\tau}_{kl}^2\upsilon_l^2)\mathbb{I}) \Big\} \prod_{n=1}^{N} p(y_n \mid f(\mathcal{W}, x_n)), \end{aligned}$$

with $\theta = \{\mathcal{W}, \mathcal{T}, \kappa^2, \rho_\kappa, c^2\}$, $\mathcal{T} = \{\{\tau_{kl}^2\}_{k=1,l=1}^{K,L}, \{\upsilon_l^2\}_{l=1}^{L}, \{\lambda_{kl}\}_{k=1,l=1}^{K,L}, \{\vartheta_l\}_{l=1}^{L}\}$.

### 3.4. Non-Centered Parameterization

Both the standard and regularized horseshoe priors exhibit strong correlations between the weights $w_{kl}$ and the scales $\tau_{kl}\upsilon_l$. While their favorable sparsity inducing properties stem from this coupling, it also gives rise to coupled posteriors that exhibit pathological funnel shaped geometries (Betancourt and Girolami, 2015; Ingraham and Marks, 2017) that are difficult to reliably sample or approximate. Fully factorized approximations are particularly problematic and can lead to non-sparse solutions erasing the benefits of using the horseshoe prior.

Adopting non-centered parameterizations (Ingraham and Marks, 2017), helps alleviate the issue. Consider a reformulation of Equation 2,

$$\beta_{kl} \sim \mathcal{N}(0, \mathbb{I}), \quad w_{kl} = \tilde{\tau}_{kl}\upsilon_l\beta_{kl}, \tag{3}$$

where the distribution on the scales are left unchanged. Since the scales and weights are sampled from independent prior distributions and are *marginally* uncorrelated, such a parameterization is referred to as non-centered. The likelihood is now responsible for introducing the coupling between the two, when conditioning on observed data. Non-centered parameterizations are known to lead to simpler posterior geometries (Betancourt and Girolami, 2015). Figure 3 summarizes the conditional dependencies assumed by the centered and the non-centered horseshoe BNN models. As with the Half-Cauchy reparameterization, this parameterization does not change the functional form of the standard or regularized horseshoe prior, but it will improve our ability to perform inference.

## 4. Variational Inference for Model Learning

We approximate the intractable posterior $p(\theta \mid \mathcal{D})$ with a computationally convenient alternative. The first step involves selecting a tractable family of distributions $q(\theta \mid \phi)$, with free variational parameters $\phi$. Next, learning involves optimizing $\phi$ such that the Kullback-Liebler divergence between the approximation and the true posterior, $\mathrm{KL}(q(\theta \mid \phi)||p(\theta \mid \mathcal{D}))$ is minimized. This is equivalent to maximizing the lower bound to the marginal likelihood (or evidence) $p(\mathcal{D})$, $p(\mathcal{D}) \geq \mathcal{L}(\phi) = \mathbb{E}_{q(\theta|\phi)}[\ln\, p(\mathcal{D}, \theta)] + \mathbb{H}[q(\theta \mid \phi)]$.

### 4.1. Variational Approximation Choices

The choice of the approximating family governs the quality of inference. The more flexible the approximating family the better it approximates the true posterior. Below, we first describe a straight-forward fully-factored approximation and then more sophisticated structured approximations that we demonstrate to have better statistical properties.

#### 4.1.1. Fully Factorized Approximation

The simplest possibility is to use a fully factorized variational family,

$$q(\theta \mid \phi) = \prod_{a \in \{c, \kappa, \rho_\kappa\}} q(a \mid \phi_a) \prod_{i,j,l} q(\beta_{ij,l} \mid \phi_{\beta_{ij,l}}) \prod_{k,l} q(\tau_{kl}^2 \mid \phi_{\tau_{kl}}) q(\lambda_{kl} \mid \phi_{\lambda_{kl}}) \prod_{l} q(\upsilon_l^2 \mid \phi_{\upsilon_l}) q(\vartheta_l \mid \phi_{\vartheta_l}). \tag{4}$$

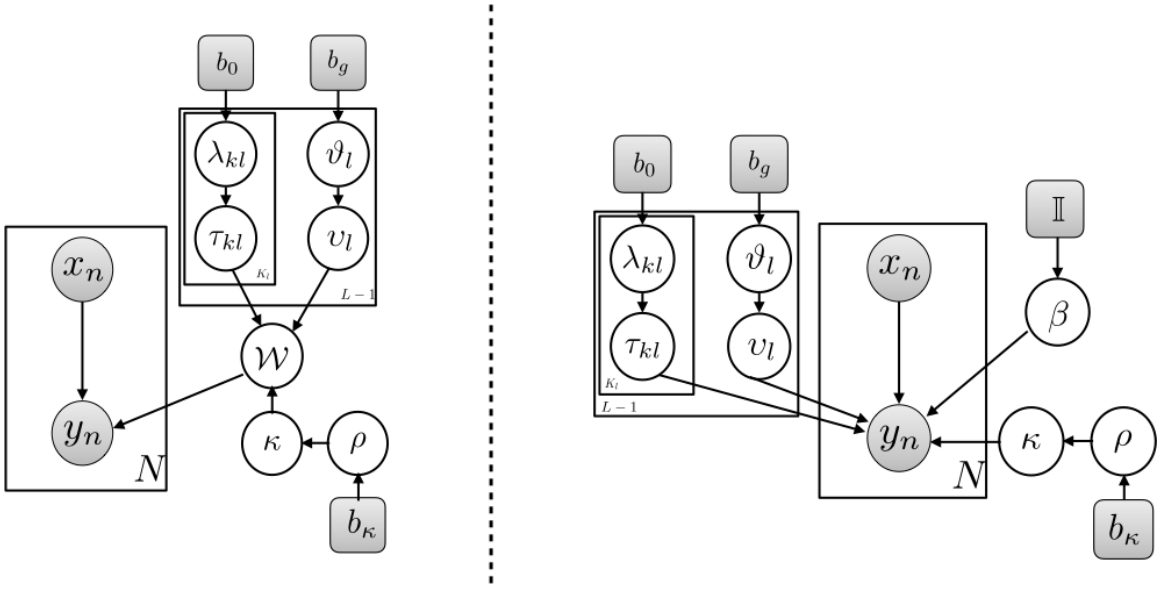


Figure 3: **Graphical models** capturing the conditional dependencies assumed by Bayesian Neural Networks with horseshoe priors. Left: Centered parameterization, Right: Non-centered parameterization necessary for robust inference.

Restricting the variational distribution for the non-centered weight $\beta_{ij,l}$ between units $i$ in layer $l-1$ and $j$ in layer $l$, $q(\beta_{ij,l} \mid \phi_{\beta_{ijl}})$ to the Gaussian family $\mathcal{N}(\beta_{ij,l} \mid \mu_{ij,l}, \sigma^2_{ij,l})$, and the non-negative scale parameters $\tau^2_{kl}$ and $\upsilon^2_l$ and the variance of the output layer weights to the log-Normal family, $q(\ln \tau^2_{kl} \mid \phi_{\tau_{kl}}) = \mathcal{N}(\mu_{\tau_{kl}}, \sigma^2_{\tau_{kl}})$, $q(\ln \upsilon^2_l \mid \phi_{\upsilon_l}) = \mathcal{N}(\mu_{\upsilon_l}, \sigma^2_{\upsilon_l})$, and $q(\ln \kappa^2 \mid \phi_\kappa) = \mathcal{N}(\mu_\kappa, \sigma^2_\kappa)$, allows for the development of straightforward inference algorithms (Ghosh and Doshi-Velez, 2017a; Louizos et al., 2017). While other variational approximations for the scale parameters are possible, the log-Normal family is particularly convenient. It provides a simple parameterization allowing for reparameterized gradients, it is closed under multiplication, the product of global and local scales is thus another log-Normal distribution, and as we will see later in this section, it allows for convenient modeling of correlations between weights and scales. It is not necessary to impose distributional constraints on the variational approximations of the auxiliary variables $\vartheta_l$, $\lambda_{kl}$, or $\rho_\kappa$. Conditioned on the other variables, the optimal variational family for these latent variables follow inverse Gamma distributions. We refer to this approximation as the *factorized* approximation.

*Parameter-tying in the fully-factorized approximation.* The conditional variational distribution on $w_{kl}$ implied by Equations 3, 4 is $q(w_{kl} \mid \tau_{kl}, \upsilon_l) = \mathcal{N}(w_{kl} \mid \tau_{kl}\upsilon_l\mu_{kl}, (\tau_{kl}\upsilon_l)^2\Psi)$, where $\Psi$ is a diagonal matrix with elements populated by $\sigma^2_{ij,l}$ and $\mu_{kl}$ consists of the corresponding variational means $\mu_{ij,l}$. The distributions of weights incident into a unit are thus coupled through $\tau_{kl}\upsilon_l$ while all weights in a layer are coupled through the layer wise scale $\upsilon_l$. This view suggests that using a simpler approximating family $q(\beta_{ij,l} \mid \phi_{\beta_{ijl}}) = \mathcal{N}(\beta_{ij,l} \mid \mu_{ij,l}, 1)$ results in an isotropic Gaussian approximation $q(w_{kl} \mid \tau_{kl}, \upsilon_l) = \mathcal{N}(w_{kl} \mid$

$\tau_{kl}\upsilon_l\mu_{kl}, (\tau_{kl}\upsilon_l)^2\mathbb{I})$. Crucially, the scale parameters $\tau_{kl}\upsilon_l$ still allow for pruning of units when the scales approach zero. Moreover, by tying the variances of the non-centered weights together this approximation effectively halves the number of variational parameters to be learned and stored. Variational parameters associated with the network weights grow quadratically with network width while the scale and auxiliary variable parameters grow linearly. Hence, storage and learning costs in large networks are dominated by the weight parameters.

We find in Section 6.5 that such parameter tying speeds up training. However, this computational benefit comes at a price. The tied approximation retains the model selection benefits, but the variational distributions for the units that escape shrinkage are forced to be isotropic. This can lead to poorer predictive performance on regression tasks.

#### 4.1.2. Structured Variational Approximations

Although computationally convenient, the factorized approximations fail to capture posterior correlations among the network weights, and more pertinently for the horseshoe prior, between weights and their scales.

*Capturing layer-wise weight correlations* We take a step towards a more structured variational approximation by using a layer-wise matrix variate normal (Gupta and Nagar, 1999) variational distribution for the non-centered weights while retaining the form of the other factors from Equation 4. Let $\beta_l \in \mathbb{R}^{(K_{l-1}+1)\times K_l}$ denote the set of weights between layers $l-1$ and $l$, then under this variational approximation we have, $q(\beta_l \mid \phi_{\beta_l}) = \mathcal{MN}(\beta_l \mid M_l, U_l, V_l)$, where $M_l \in \mathbb{R}^{(K_{l-1}+1)\times K_l}$ is the mean, $V_l \in \mathbb{R}^{K_l \times K_l}$ and $U_l \in \mathbb{R}^{(K_{l-1}+1)\times(K_{l-1}+1)}$ capture the covariances among the columns and rows of $\beta_l$, thereby modeling dependencies among the variational approximation to the weights in a layer. Louizos and Welling (2016) demonstrated that even when $U_l$ and $V_l$ are restricted to be diagonal, the matrix Gaussian approximation can lead to significant improvements over fully factorized approximations for standard BNNs. We call this the *semi-structured* approximation because it only captures correlations among weights but not between weights and scales.

*Capturing weight-scale correlations* The horseshoe prior exhibits strong correlations between weights and their scales, which encourages strong posterior coupling between $\beta_{kl}$ and $\tau_{kl}$. However, the variational families considered to this point do not account for this dependency. For effective shrinkage towards zero, it is important that the variational approximations are able to capture this strong dependence. We begin by defining, $B_l = \begin{bmatrix} \beta_l \\ \nu_l^T \end{bmatrix}$, $\nu_l = [\nu_{1l}, \ldots, \nu_{K_l l}]^T$, and $\nu_{kl} = \ln \tau_{kl}^2$. Using the variational approximation $q(B_l \mid \phi_{B_l}) = \mathcal{MN}(B_l \mid M_l, U_l, V_l)$, allows us to retain the coupling between weights incident onto a unit and the corresponding unit specific scales, with appropriate parameterizations of $U_l$. In particular, we note that a diagonal $U_l$ fails to capture the necessary correlations, and defeats the purpose of using a matrix Gaussian variational family to model the posterior of $B_l$. To retain computational efficiency while capturing dependencies among the rows of $B_l$ we enforce a low-rank structure, $U_l = \Psi_l + h_l h_l^T$, where $\Psi_l \in \mathbb{R}^{(K_{l-1}+2)\times(K_{l-1}+2)}$ is a diagonal matrix and $h_l \in \mathbb{R}^{(K_{l-1}+2)\times 1}$ is a column vector. We retain a diagonal structure for $V_l \in \mathbb{R}^{K_l \times K_l}$. We refer to this as the *structured* approximation. In Section 6, we find that this structured approximation indeed leads to stronger shrinkage towards zero in

Table 1: Variational Approximation Families.

| Approximation | Description |
|---|---|
| Factorized | $q(\nu_l \mid \phi_{\nu_l})q(\beta_l \mid \phi_{\beta_l}) = \prod_{i,j,l} \mathcal{N}(\mu_{ij,l}, \sigma^2_{ij,l}) \prod_{k,l} q(\nu_{kl} \mid \phi_{\nu_{kl}})$ |
| Factorized (tied) | $q(\nu_l \mid \phi_{\nu_l})q(\beta_l \mid \phi_{\beta_l}) = \prod_{i,j,l} \mathcal{N}(\mu_{ij,l}, 1) \prod_{k,l} q(\nu_{kl} \mid \phi_{\nu_{kl}})$ |
| Semi-structured | $q(\nu_l \mid \phi_{\nu_l})q(\beta_l \mid \phi_{\beta_l}) = \mathcal{MN}(\beta_l \mid M_l, U_l, V_l) \prod_{k,l} q(\nu_{kl} \mid \phi_{\nu_{kl}})$ |
| Structured | $q(\beta_l, \nu_l \mid \phi_{B_l}) = \mathcal{MN}(B_l \mid M_l, U_l, V_l)$ |

the recovered solutions. When combined with an appropriate pruning rule, it significantly compresses networks with excess capacity.

Table 1 summarizes the variational approximations introduced in this section. All our variational approximations employ a, possibly correlated, Gaussian distribution for the weights and log-Normal distributions for the scale parameters. These choices were primarily governed by computational ease. The Gaussian distribution on weights allows for the use of the "local reparameterization trick" (Kingma et al., 2015), and consequently low variance gradients. The log-Normal distributions are similarly amenable to reparameterized gradients again allowing for low variance gradients. Moreover, it allows us to easily model correlations between the weights and the log scales through a multivariate Gaussian distribution. Other choices of variational distributions are certainly possible. Exploring variational distribution families that better trade off computational ease for accuracy is an interesting direction for future work.

### 4.2. Black Box Variational Inference

Irrespective of the chosen variational family, the resulting evidence lower bound (ELBO),

$$\mathcal{L}(\phi) = \sum_n \mathbb{E}_{q(\theta|\phi)}[\ln\, p(y_n \mid f(\theta, x_n))] + \mathbb{E}_{q(\theta|\phi)}[\ln\, p(\theta \mid b_0, b_g, b_\kappa, c_a, c_b)] + \mathbb{H}[q(\theta \mid \phi)], \quad (5)$$

is challenging to evaluate. Here we have used $\beta$ to denote the set of all non-centered weights in the network. The non-linearities introduced by the neural network and the potential lack of conjugacy between the neural network parameterized likelihoods and the horseshoe priors render the first expectation in Equation 5 intractable. Consequently, the traditional prescription of optimizing the ELBO by cycling through a series of fixed point updates is no longer available.

Recent progress in black box variational inference (Kingma and Welling, 2014; Rezende et al., 2014; Ranganath et al., 2014; Titsias and Lázaro-gredilla, 2014) subverts this difficulty. These techniques compute noisy unbiased estimates of the gradient $\nabla_\phi \hat{\mathcal{L}}(\phi)$, by approximating the offending expectations with unbiased Monte-Carlo estimates and relying on either score function estimators (Williams, 1992; Ranganath et al., 2014) or reparameterization gradients (Kingma and Welling, 2014; Rezende et al., 2014; Titsias and Lázaro-gredilla, 2014) to differentiate through the sampling process. With the unbiased gradi-

ents in hand, stochastic gradient ascent can be used to optimize the ELBO. In practice, reparameterization gradients exhibit significantly lower variances than their score function counterparts and are typically favored for differentiable models. The reparameterization gradients rely on the existence of a parameterization that separates the source of randomness from the parameters with respect to which the gradients are sought. For our Gaussian variational approximations, the well known non-centered parameterization, $\zeta \sim \mathcal{N}(\mu, \sigma^2) \Leftrightarrow \epsilon \sim \mathcal{N}(0,1), \zeta = \mu + \sigma\epsilon$, allows us to compute Monte-Carlo gradients,

$$\nabla_{\mu,\sigma}\mathbb{E}_{q_w}[g(w)] \Leftrightarrow \nabla_{\mu,\sigma}\mathbb{E}_{\mathcal{N}(\epsilon|0,1)}[g(\mu+\sigma\epsilon)] \approx \frac{1}{S}\sum_s \nabla_{\mu,\sigma} g(\mu + \sigma\epsilon^{(s)}), \tag{6}$$

for any differentiable function $g$ and $\epsilon^{(s)} \sim \mathcal{N}(0,1)$. Furthermore, practical implementations of variational Bayesian neural networks typically use a further re-parameterization to lower variance of the gradient estimator. They sample from the implied variational distribution over a layer's pre-activations instead of directly sampling the much higher dimensional weights (Kingma et al., 2015).

#### 4.2.1. Variational distribution on pre-activations for structured approximating families

While the "local" re-parametrization is straightforward for the factorized and semi-structured approximations, it is more involved for the structured case. We observe that by factorizing $q(B_l \mid \phi_{B_l})$ as $q(\beta_l \mid \nu_l, \phi_{\beta_l})q(\nu_l \mid \phi_{\nu_l})$ and that conditioned on $\nu_l \sim q(\nu_l \mid \phi_{\nu_l})$, $\beta_l$ follows another matrix Gaussian distribution. This conditional variational distribution is given by, $q(\beta_l \mid \nu_l, \phi_{\beta_l}) = \mathcal{MN}(M_{\beta_l|\nu_l}, U_{\beta_l|\nu_l}, V)$. The conditional mean $M_{\beta_l|\nu_l}$ and row variances $U_{\beta_l|\nu_l}$ expressions are derived in the supplement. It then follows that $b = \beta_l^T a$ for an input $a \in \mathbb{R}^{(K_{l-1}+1)\times 1}$ into layer $l$, is distributed as,

$$b \mid a, \nu_l, \phi_{\beta_l} \sim \mathcal{N}(b \mid \mu_b, \Sigma_b), \tag{7}$$

with $\mu_b = M^T_{\beta_l|\nu_l} a$, and $\Sigma_b = (a^T U_{\beta_l|\nu_l} a)V$. Since, $a^T U_{\beta_l|\nu_l} a$ is scalar and $V$ is diagonal, $\Sigma$ is diagonal as well. For regularized HS-BNN, recall that the pre-activation of node $k$ in layer $l$, is $u_{kl} = \tilde{\tau}_{kl}\upsilon_l b$, and the corresponding variational posterior is,

$$q(u_{kl} \mid \mu_{u_{kl}}, \sigma^2_{u_{kl}}) = \mathcal{N}(u_{kl} \mid \mu_{u_{kl}}, \sigma^2_{u_{kl}}); \quad \mu_{u_{kl}} = \tilde{\tau}^{(s)}_{kl}\upsilon^{(s)}_l \mu_{bk}; \quad \sigma^2_{u_{kl}} = \tilde{\tau}^{(s)^2}_{kl}\upsilon^{(s)^2}_l \Sigma_{bk,k}, \tag{8}$$

where $\tau^{(s)}_{kl}$, $\upsilon^{(s)}_l, c^{(s)}$ are samples from the corresponding log-Normal posteriors and $\tilde{\tau}^{(s)}_{kl}$ is constructed as $c^{(s)^2}\tau^{(s)^2}_{kl}/(c^{(s)^2} + \tau^{(s)^2}_{kl}\upsilon^{(s)^2}_l)$. Thus, to compute the pre-activation distributions under the structured variational approximation, we first sample the vector $\nu_l$ and then sampling the pre-activations conditioned on $\nu_l$ and inputs into the layer. Crucially, we can perform all operations without having to sample the high dimensional weights $\beta_l$.

#### 4.2.2. Algorithm

We now have a simple prescription for optimizing Equation 5. Recursively sampling the variational posterior of Equation 8 for each layer of the network, allows us to forward

propagate information through the network. Using the reparameterizations (Equation 6), allows us to differentiate through the sampling process. We compute the necessary gradients through reverse mode automatic differentiation tools (Maclaurin et al., 2015). With the gradients in hand, we optimize $\mathcal{L}(\phi)$ with respect to the variational weights $\phi_B$, per-unit scales $\phi_{\tau_{kl}}$, per-layer scales $\phi_{\upsilon_l}$, and the variational scale for the output layer weights, $\phi_\kappa$ using Adam (Kingma and Ba, 2015). For the structured approximation, we found a further heuristic — wherein we restrict the variational distributions such that the parameters $\Psi$, $V$, and $h$ lie in the $\ell_2$ unit-ball produced better predictive performance. A more rigorous treatment of this heuristic is left as future work. Conditioned on these, the optimal variational posteriors of the auxiliary variables $\vartheta_l$, $\lambda_{kl}$, and $\rho_\kappa$ follow inverse gamma distributions. Fixed point updates that maximize $\mathcal{L}(\phi)$ with respect to $\phi_{\vartheta_l}, \phi_{\lambda_{kl}}, \phi_{\rho_\kappa}$, holding the other variational parameters fixed are available. It can be shown that,

$$q(\lambda_{kl} \mid \phi_{\lambda_{kl}}) = \text{Inv-Gamma}(\lambda_{kl} \mid 1, \mathbb{E}[\frac{1}{\tau_{kl}^2}] + \frac{1}{b_0^2}). \tag{9}$$

The distributions of the other auxiliary variables are analogous. By alternating between gradient and fixed point updates to maximize the ELBO in a coordinate ascent fashion we learn all variational parameters jointly (see Algorithm 1).

#### 4.2.3. Computational Considerations

The primary computational bottleneck for the structured approximation arises in computing the pre-activations in Equation 7. While computing $\Sigma_b$ in the factorized approximation involves a single inner product, in the structured case it requires the computation of the quadratic form $a^T U_{M_{\beta_l|\nu_l}} a$ and a point wise multiplication with the elements of $V_l$. Owing to the diagonal plus rank-one structure of $U_{M_{\beta_l|\nu_l}}$, we only need two inner products, followed by a scalar squaring and addition to compute the quadratic form and $K_l$ scalar multiplications for the point-wise multiplication with $V_l$. Thus the structured approximation is only marginally more expensive. Further, it uses only $(K_l + 2) \times (K_{l-1} + 1)$ weight variance parameters per layer, instead of $K_l \times (K_{l-1} + 1)$ parameters used by the factorized approximation. Not having to compute gradients and update these additional parameters further mitigates the performance difference.

**Algorithm 1** Training with Standard and Regularized HS-BNNs

**Input** Model $p(\mathcal{D}, \theta)$, variational approximation $q(\theta \mid \phi)$, number of iterations T.
**Output**: Variational parameters $\phi$
Initialize variational parameters $\phi$.
**for** `T iterations` **do**
  Update $\phi_c$, $\phi_\kappa$, $\phi_\gamma$, $\{\phi_{B_l}\}_l$, $\{\phi_{\upsilon_l}\}_l \leftarrow \text{ADAM}(\mathcal{L}(\phi))$.
  **for** `all hidden layers` $l$ **do**
    Conditioned on $\phi_{B_l}$, $\phi_{\upsilon_l}$ update $\phi_{\vartheta_l}$, $\phi_{\lambda_{kl}}$ using fixed point updates (Equation 9).
  **end for**
  Conditioned on $\phi_\kappa$ update $\phi_{\rho_\kappa}$ via the corresponding fixed point update.
**end for**

### 4.3. Pruning Rule

The group horseshoe and its regularized variant provide strong shrinkage towards zero for small $w_{kl}$. However, since we infer a posterior distribution the shrunk weights, although tiny, are never actually zero. A user-defined thresholding rule is required to prune away the shrunk weights. In previous work, Louizos et al. (2017) first summarized the inferred posterior distributions using a point estimate and then used the point summary to define a thresholding rule. Here, we propose an alternate thresholding rule that does not rely on a point summary of the posterior. We prune away a unit if $p(\tau_{kl}\upsilon_l < \delta) > p_0$, that is, when the probability of the scale being small is sufficiently large. The scale $\delta$ and the probability $p_0$ are user defined parameters, with $\tau_{kl} \sim q(\tau_{kl} \mid \phi_{\tau_{kl}})$ and $\upsilon_l \sim q(\upsilon_l \mid \phi_{\upsilon_l})$. To see why this rule is sensible, recall that for units which experience strong shrinkage the regularized horseshoe tends to the horseshoe. Under the horseshoe prior, $\tau_{kl}\upsilon_l$ governs the (non-negative) scale of the weight node vector $w_{kl}$. Therefore, under our thresholding rule, we prune away nodes whose posterior scales, place probability greater than $p_0$ below a sufficiently small threshold $\delta$. In our experiments, we set $p_0 = 0.9$ and $\delta$ to either $1e-3$ or $1e-5$ (for the year data set). Further, under our variational approximations, $\tau_{kl}\upsilon_l$ follows a log-Normal distribution. Evaluating the thresholding rule is as simple as evaluating the log-Normal cumulative distribution.

## 5. Related Work

Early work on Bayesian neural networks can be traced back to (Buntine and Weigend, 1991; MacKay, 1992; Neal, 1993). Neal (1993) introduced Hamiltonian Monte Carlo (HMC) for exploring the posterior over network weights, while MacKay (1992); Buntine and Weigend (1991) relied on Laplace approximation to characterize the network posterior. HMC remains the gold standard for posterior inference in BNNs. However, it does not scale well to modern architectures or the large data sets required to learn them. Similarly, vanilla application of the Laplace approximation has difficulty scaling to modern architectures with millions of parameters. Recent advances in stochastic MCMC methods (Welling and Teh, 2011; Chen et al., 2014) and stochastic variational methods (Blundell et al., 2015; Rezende et al., 2014), black-box variational inference and alpha-divergence minimization (Hernandez-Lobato et al., 2016; Ranganath et al., 2014), probabilistic backpropagation (Hernández-Lobato and Adams, 2015) as well as scalable Laplace approximations (Ritter et al., 2018) have reinvigorated interest in BNNs by allowing efficient scalable inference. We direct the interested reader to Yao et al. (2019) for a thorough comparison of modern inference algorithms for BNNs.

Work on learning structure in BNNs has received less attention. Blundell et al. (2015) introduce a mixture-of-Gaussians prior on the weights, with one mixture tightly concentrated around zero, thus approximating a spike and slab prior over weights. Others (Kingma et al., 2015; Gal and Ghahramani, 2016) have noticed connections between Dropout and approximate variational inference (Srivastava et al., 2014). In particular, Molchanov et al. (2017) show that the interpretation of Gaussian dropout as performing variational inference in a network with log uniform priors over weights leads to sparsity in weights. The goal of turning off edges is very different than the approach considered here, which performs model selection over the appropriate number of nodes. Yet others (Neklyudov et al., 2017)

have proposed pruning units via truncated log-normal priors over unit scales. However, they do not place priors over network weights and are unable to infer posterior uncertainty over weights, which may lead to poorer predictive uncertainties. Related but orthogonal research (Adams et al., 2010; Song et al., 2017) has focused on the problem of structure learning in deep belief networks.

Closely related to this work is the work of Louizos et al. (2017). They also consider group Horseshoe priors for model compression. In this paper, we take a closer look at the model selection properties afforded by such priors. We also provide robust extensions to the model—via the regularized Horseshoe prior—the inference—via structured variational approximations—and provide a thresholding rule to prune units with small scales.

There is also a body of work on learning structure in non-Bayesian neural networks. Early work (LeCun et al., 1990; Hassibi et al., 1993) pruned networks by analyzing second-order derivatives of the objectives. More recently, Wen et al. (2016) describe applications of structured sparsity not only for optimizing filters and layers but also computation time. Closer to our work in spirit, Ochiai et al. (2016); Scardapane et al. (2017); Alvarez and Salzmann (2016); Murray and Chiang (2015), use group sparsity to prune groups of weights—e.g. weights incident to a node. However, these approaches do not model uncertainty in weights and provide uniform shrinkage to all parameters. Our approach provides group shrinkage while retaining weight uncertainties.

## 6. Experiments

In this section, we carefully evaluate our proposed modeling and inferential contributions. First, on synthetic data we explore the properties of the horseshoe BNN (HS-BNN) and its different parameterizations. We find that employing a non-centered parameterization is necessary for effective model selection. Next, on several real-world data sets we vet both the model selection effects and the predictive performance of the non-centered HS-BNN and find that it is able to provide model selection benefits without significantly sacrificing predictive performance. We next evaluate the impact of our modeling and inference extensions. On the modeling front, we find that the regularized Horseshoe BNN (reg-HS) improves upon the predictive performance of HS-BNN, especially on smaller data sets. Through further controlled comparisons we find that the structured variational approximation improves upon the factorized variants by recovering solutions that exhibit significantly stronger shrinkage towards zero. Finally, we end with experiments that go beyond prediction and demonstrate the value of HS-BNNs for reinforcement learning. In our experiments, unless otherwise mentioned we use a learning rate of 0.005. For regression problems we employ a Gaussian likelihood with an unknown precision $\gamma$, $p(y_n|f(\mathcal{W}, x_n), \gamma) = N(y_n|f(\mathcal{W}, x_n), \gamma^{-1})$. We place a vague prior on the precision, $\gamma \sim \text{Gamma}(6, 6)$ and approximate the posterior over $\gamma$ using another Gamma distribution. For classification problems we use a Categorical distribution parameterized by $\mathcal{S}(f(\mathcal{W}, x_n))$, where $\mathcal{S}$ is the softmax transformation.

### 6.1. Underfitting variational Bayesian neural networks

We began by exploring the effect of model specification on BNNs learned via fully factorized variational inference. We generated varying amounts of synthetic data from $y \sim \mathcal{N}(sin(x), 0.1)$ and fit fully connected BNN models of varying size. We used ReLU activa-

tions and placed factorized Gaussian priors over the network weights, $w_{ij,l} \sim \mathcal{N}(0, 1)$, where $w_{ij,l}$ denotes the weight between nodes $i$ in layer $l-1$ and $j$ in layer $l$. We used black-box variational inference with locally reparameterized gradients (Kingma et al., 2015). We trained the models to convergence from thirty random restarts and selected the highest evidence lower bound solution.

We found that BNNs trained under this regime were not robust to model specification, quickly increasing the uncertainty of their predictions as more nodes were added to the network while keeping the amount of training data fixed. This is illustrated in Figure 1. Figure 4 visualizes the corresponding weights, where we plot the 2-Wasserstein distance between the inferred variational posterior and the prior, $\mathcal{W}_2(\mathcal{N}(w_{ij,l} \mid \mu_{ij,l}, \sigma^2_{ij,l}), \mathcal{N}(w_{ij,l} \mid 0, 1))$. This experiment leads to the following observation. *A sufficiently wide BNN with $\mathcal{N}(0, \sigma^2)$ prior on weights, when conditioning on a sufficiently small data set and trained via fully factorized variational inference produces a posterior approximation that tends towards the prior and consequently a posterior predictive distribution that reverts to the prior predictive distribution.* To gain further insight into this observation, note that the ELBO for a BNN is $\sum_{n=1}^{N} \mathbb{E}_{q(\mathcal{W}|\phi)}[\ln p(y_n \mid f(\mathcal{W}, x_n)] - \mathrm{KL}(q(\mathcal{W} \mid \phi)||p(\mathcal{W})) = \sum_{n=1}^{N} \mathbb{E}_{q(\mathcal{W}|\phi}[\ln p(y_n \mid f(\mathcal{W}, x_n)] - \sum_{j=1}^{P} \mathrm{KL}(q(w_j \mid \phi)||\mathcal{N}(w_j \mid 0, 1))$, where the equality follows from the fully factorized variational approximation. When $P$, the number of weights, is significantly larger than $N$, the KL term in the ELBO dominates incentivizing the posterior approximation $q(\mathcal{W} \mid \phi)$ to revert to the prior.

### 6.2. Demonstrations of HS-BNN Behavior on Synthetic Data

In this section, we first demonstrate the value of the Horseshoe prior by returning to the the demonstration in Figure 1. We also demonstrate the importance of non-centered parameterization of the Horseshoe distribution.

#### 6.2.1. Horseshoe priors demonstrate robustness to model specification.

In Figure 5, we reproduce the results of Figure 1 alongside the fit of a 1000 unit HS-BNN. While the fit of the BNN with zero mean, unit variance Gaussian priors quickly deteriorates with increasing capacity conditioned on the limited amount of training data, HS-BNN is robust to model-misspecification, pruning away the additional capacity. Like the BNN, HS-BNN training employed thirty random initializations. Solutions with the highest evidence lower bound are plotted.

To verify that this effect holds across data sets, we also reproduced the noisy polynomial regression problem considered in (Hernández-Lobato and Adams, 2015), $y_n = x_n^3 + \epsilon_n$, $\epsilon_n \sim \mathcal{N}(0, 9)$. Figure 6 illustrates the fits recovered by a Bayesian neural network with Gaussian priors against a HS-BNN conditioned on data sampled from $\mathcal{U}(-4, +4)$. We sample a total of 500 points, using a 100 for training and rest for testing. We again find that the BNN underfits the data and significantly overestimates the predictive uncertainty. HS-BNN on the other hand provides improved estimates of predictive uncertainty. We also plot the test predictive log likelihoods averaged over five random trials in Figure 6. Observe that with increasing size, the predictive likelihoods drop significantly for BNN. HS-BNN exhibits robustness with more modest drops in performance. Later in this section we will

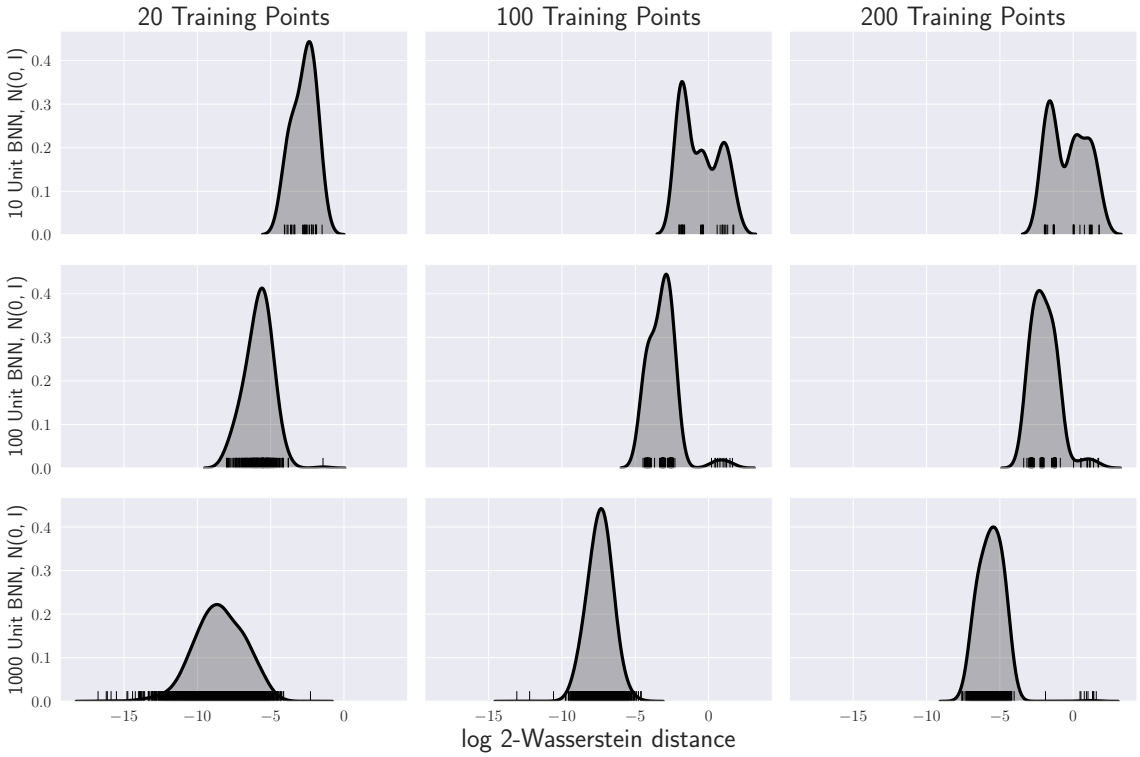


Figure 4: **Analysis of posterior weights of variational BNNs with $\mathcal{N}(0,1)$ priors**. This figure provides the posterior weights corresponding to the predictive distributions presented in Figure 1. Each rug represents the (squared) 2-Wasserstein distance the inferred posterior and the prior, $\mathcal{W}_2(\mathcal{N}(w_{ij,l} \mid \mu_{ijl}, \sigma^2_{ijl}), \mathcal{N}(w_{ij,l} \mid 0, 1))^2$ for weight $w_{ij,l}$. A kernel density estimate of the Wasserstein distances are also plotted. Along each column, from top to bottom the width of the network increases while the number of training instances remains constant. Observe that the density plots exhibit a leftward shift from top to bottom along each column and a rightward shift along each row from left to right. These results indicate that for a fixed size network as the number of training instances increase posterior weights deviate more strongly from the prior and consequently exhibit better fits to the data. On the other hand for a fixed amount of training data, increasing the network size encourages the prior to overwhelm the data resulting in posteriors that do not deviate from the prior. *Note that with respect to Figure 1, we have transposed the columns and rows here to more clearly illustrate the effects of network size on posterior inference.*

see that using more sophisticated variational approximations further improves results where predictive performance no longer deteriorates with increasing capacity.

#### 6.2.2. Non-centered parameterizations are crucial for reliable inferences

The results above crucially rely on using a non-centered parameterization. We illustrate the importance of the non-centered parameterization with a simple two dimensional classification problem generated by sampling data uniformly at random from $[-1, +1] \times [-1, +1]$

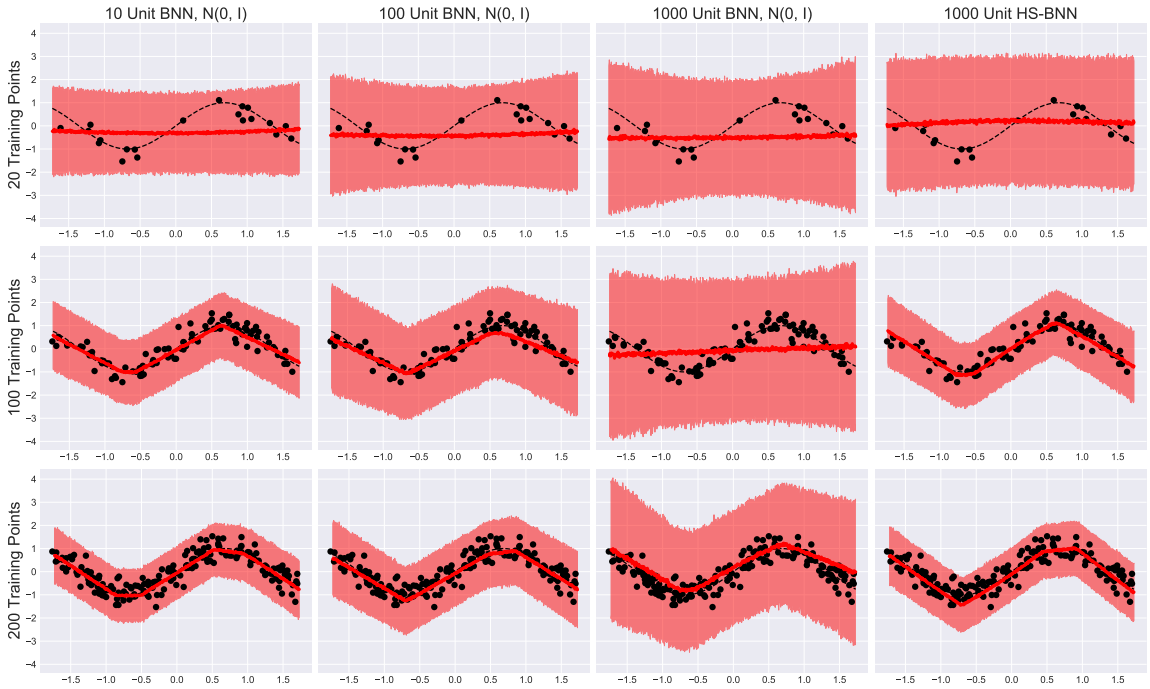


Figure 5: **Horseshoe BNN**. We return to the example from Figure 1. The 1000 node HS-BNN (right-most column) underfits significantly less than its 1000 node standard BNN counterpart.

and using a 2-2-1 network, whose parameters are known *a-priori* to generate the class labels. We train three Bayesian neural networks with a 15-unit hidden layer on these data, with Gaussian priors, with non-centered horseshoe priors, and with centered horseshoe priors. All three models are able to easily fit the data and provide high predictive accuracy. However, the structure learned by the three models are very different. Figure 7 plots the posterior means of weight vectors ($\mathbb{E}[w_{kl}]$) incident onto a unit. Unsurprisingly, the BNN with Gaussian priors does not exhibit shrinkage towards zero. In contrast, models employing the horseshoe prior are able to prune units away by setting all incident weights to tiny values. It is interesting to note that even for this highly stylized example the centered parameterization struggles to recover the true structure of the underlying network. In contrast, the non-centered parameterization prunes away all but two units. The effect remains even with 100 units in the hidden layer (Figure 18) in Appendix C.2.

### 6.3. Evaluation of HS-BNN Performance on Real Data Sets

In Section 6.2, we demonstrated the basic properties of the HS BNN and the importance of the non-centered parameterization. We now turn to the evaluation of the HS BNN on real data sets. In the experiments below, we use the fully factorized variational approximation, unless otherwise stated; we will return to comparisons between different variational approximations in Section 6.5.

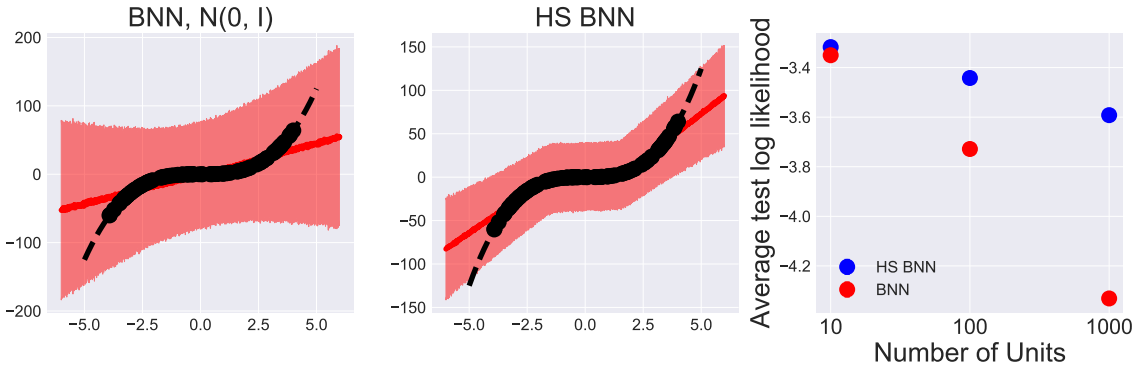


Figure 6: **Noisy polynomial regression** $y_n = x_n^3 + \epsilon_n$, $\epsilon_n \sim \mathcal{N}(0, 9)$. The crosses indicate training data and the dotted line visualizes the noise free cubic function, and in red we visualize the predicted means and $\pm$ 3 standard deviations for single layer networks with 1000 hidden units. Right: Average predictive log likelihood for single layer networks with 10, 100 and 1000 units. The error bars are smaller than the size of the plotted markers. HS-BNN improves upon BNN and the held out log likelihoods deteriorate more slowly with increasing model capacity. In Figure 15 we will see that employing a structured variational approximation leads to even better fits with held out likelihoods no longer deteriorating with extra capacity.

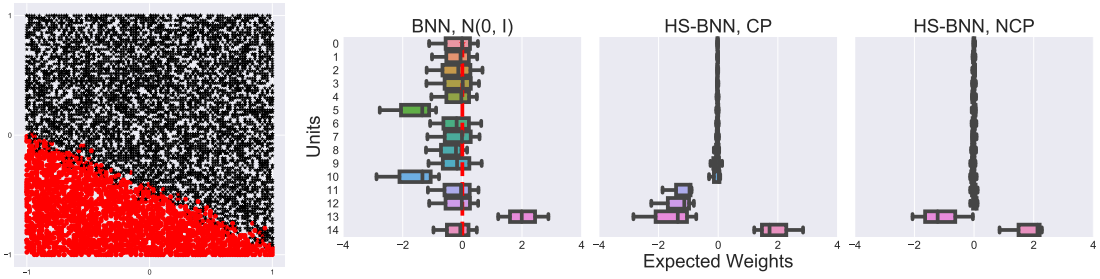


Figure 7: **Non-centered parameterization** is essential for robust inference. From left to right, A synthetic nearly linear classification problem generated from sampling a 2-2-1 network, the two classes are in red and black. Expected weights inferred with a Bayesian neural network with Gaussian priors on weights. Expected weights recovered with a centered horseshoe parameterization. Non-centered horseshoe parameterization. The box-plots display $E_q[w_{kl}]$.

#### 6.3.1. Classification with MNIST: HS-BNN learns sparse models.

We preprocessed the images in the MNIST digits data set by dividing the pixel values with 126. We explored networks with varying widths and depths all employing rectified linear units. For HS-BNN we used Adam with a learning rate of 0.005 and 500 epochs. We found that annealing in the entropy (Sønderby et al., 2016) and cross-entropy terms of the ELBO

as well as scaling the variances of the variational posterior of the weights by the number of incoming nodes improved predictive performance. We did not use a validation set to monitor performance or tune hyper-parameters.

We compare HS-BNN against modern approaches for variational learning of neural networks — variational matrix Gaussian (VMG) (Louizos and Welling, 2016), a BNN with a two-component scale mixture (SM-BNN) prior on weights proposed by Blundell et al. (2015) and a BNN with Gaussian prior (BNN) on weights. VMG uses a structured variational approximation, while the other approaches use fully factorized approximations and differ only in the type of prior used. We trained all models using the parameter settings recommended in the original papers for the competing methods.

Figure 8 summarizes our findings. We showcase results for three architectures with two hidden layers each containing 400, 800 and 1200 rectified linear hidden units. Across architectures, we find our performance to be significantly better than BNN, comparable to SM-BNN, and worse than VMG. More interestingly, we clearly see the strong shrinkage provided by the horseshoe prior. Recall that under the horseshoe prior, $w_{kl} \sim \mathcal{N}(0, \tau_{kl}^2 \upsilon_l^2 \mathbb{I})$. As the scales $\tau_{kl}\upsilon_l$ tend to zero the corresponding units (and all incident weights) are pruned away. SM-BNN also encourages shrinkage to zero, but on weights not nodes. Further, the horseshoe prior with its thicker tails and taller spike at origin encourages stronger shrinkage. To see this we compared the $\ell_2$-norms of the inferred expected weight node vectors $\mathbb{E}[w_{kl}]$ found by SM-BNN and HS-BNN. For HS-BNN the inferred scales are small for most units, with a few notable outliers that escape un-shrunk. This causes the corresponding weight vectors to be zero for the majority of units, suggesting that the model is able to effectively "turn off" extra capacity. In contrast, the weight node vectors recovered by SM-BNN (and BNN) are less tightly concentrated at zero. We also plot the density of $\mathbb{E}[w_{kl}]$ with the smallest norm in each of the three architectures. Note that with increasing architecture size (modeling capacity) the density peaks more strongly at zero, suggesting that the model is more confident in turning off the unit and not use the extra modeling capacity.

To further explore the implications of unit versus weight shrinkage, we visualize $\mathbb{E}[w_{kl}]$ learned by SM-BNN and HS-BNN in Figure 8. Weight shrinkage in SM-BNN encourages fundamentally different filters that pick up edges at different orientations. In contrast, HS-BNN's node shrinkage encourages filters that correspond to digits or superpositions of digits and may lead to more interpretable networks. Stronger shrinkage afforded by the horseshoe is again evident when visualizing filters with the lowest norms. HS-BNN filters are nearly all black when scaled with respect to the SM-BNN filters. Pruning away entire units allows for significant test-time speedups relying only on standard dense matrix operations.

Finally, in Figure 9, we provide additional results that illustrate the model selection abilities of HS-BNN. First we visualize the norms of inferred node weight vectors $\mathbb{E}[w_{kl}]$ found by BNN, SM-BNN and HS-BNN for $400-400$, $800-800$ and $1200-1200$ networks (Figure 9). Note that as we increase capacity the model selection abilities of HS-BNN becomes more obvious and as opposed to the other approaches illustrate clear inflection points and it is evident that the model is using only a fraction of its available capacity. Next, we visualize the density of the inferred node weight vectors $\mathbb{E}[w_{kl}]$ under the two models for networks $400-400$, $800-800$ and $1200-1200$. For each network we show the density of the five units with the smallest norms from either layer. Note that in all three cases HS-BNN produces weights that are more tightly concentrated around zero. Moreover for HS-BNN

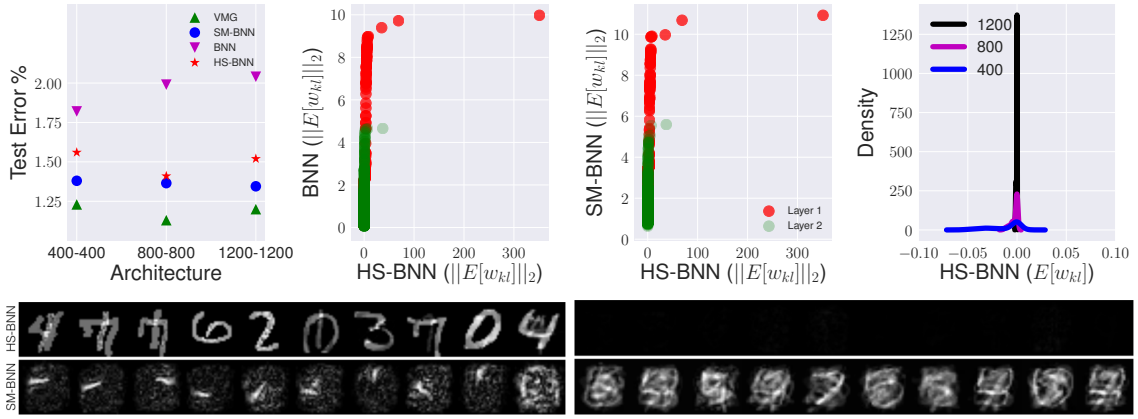


Figure 8: **MNIST Results**. TOP: *From left to right*, Test error rates for different architectures and methods. The center two plots compare the shrinkage properties of the solutions found by HS-BNN, SM-BNN and BNN. For the 1200-1200 network, we compare the expected node weight vectors inferred under the different models. We sort the recovered weight vectors $\mathbb{E}[w_{kl}]$ based on their 2-norm and compare them via scatter plots. Each circle corresponds to one of the 1200 weight node vectors. Compared to competing methods a large number of weight node vectors are zeroed out, while a small number escapes un-shrunk for HS-BNN. The rightmost plot shows the density of the unit with the lowest norm from the the three architectures. BOTTOM: $\mathbb{E}[w_{kl}]$ for the first layer. The left and right columns visualize the ten units with the largest and the smallest norms.

the concentration around zero becomes sharper with increasing modeling capacity (larger architectures), again indicating that we are pruning away additional capacity.

#### 6.3.2. Classification for Gesture Recognition: HS-BNNs again achieve shrinkage while retaining strong predictive performance.

We also experimented with a gesture recognition data set (Song et al., 2011) that consists of 24 unique aircraft handling signals performed by 20 different subjects, each for 20 repetitions. The task consists of recognizing these gestures from kinematic, tracking and video data. However, we only use kinematic and tracking data. A couple of example gestures are visualized in Figure 10. The data set contains 9600 gesture examples. A 12-dimensional vector of body features (angular joint velocities for the right and left elbows and wrists), as well as an 8 dimensional vector of hand features (probability values for hand shapes for the left and right hands) collected by Song et al. (2011) are provided as features for all frames of all videos in the data set. We additionally used the 20 dimensional per-frame tracking features made available by Song et al. (2011). We constructed features to represent each gesture by first extracting frames by sampling uniformly in time and then concatenating the per-frame features of the selected frames to produce 600-dimensional feature vectors.

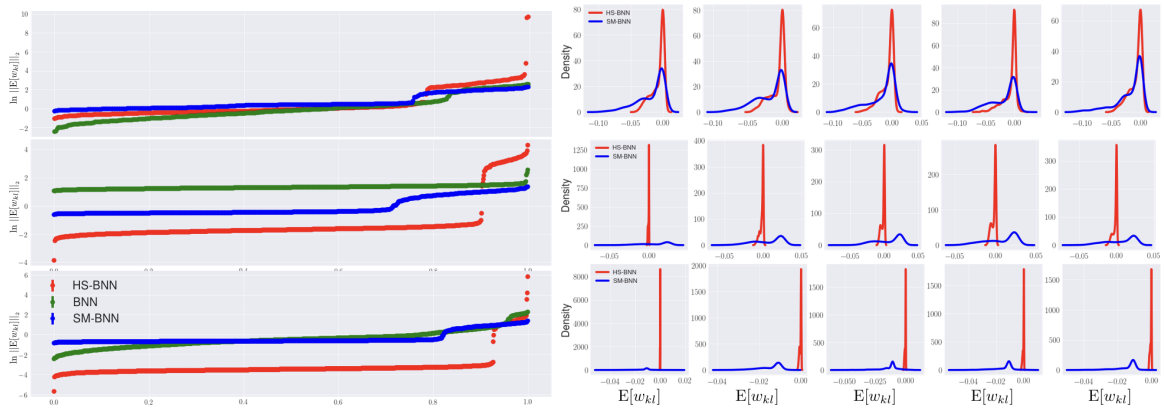


Figure 9: **Model selection in HS-BNN.** *Left*: From top to bottom, we plot log $||\mathbb{E}[w_{kl}]||_2$ for the first layer of a network with 400, 800 and 1200 units. The x-axis has been normalized by the number of units. Further exploration of sparse solutions found by HS-BNN. *Right*: Here we provide density plots for the five smallest (across all layers) node weight vectors $w_{kl}$ found by HS-BNN and SM-BNN for the 400-400 (top row), 800-800 (middle row), 1200-1200 (bottom row) network. Plots are sorted by 2-norm of $w_{kl}$, from left to right.

This is a much smaller data set than MNIST and recent work (Joshi et al., 2017) has demonstrated that a BNN with Gaussian priors performs well on this task. Figure 10 compares the performance of HS-BNN with competing methods. We train a two layer HS-BNN with each layer containing 400 units. The error rates reported are a result of averaging over five random 75/25 splits of the data. Similar to MNIST, HS-BNN significantly outperforms BNN and is competitive with VMG and SM-BNN and provides strong shrinkage.

#### 6.3.3. Regression on UCI data sets: horseshoe priors continue to learn smaller models with comparable accuracy.

We next apply our HS-BNN to various standard UCI regression data sets. We follow the experimental protocol proposed in (Hernández-Lobato and Adams, 2015) and train a single hidden layer network with 50 rectified linear units for all but the larger "Protein" and "Year" data sets for which we train a 100 unit network. For the smaller data sets we train on a randomly subsampled 90% subset and evaluate on the remainder and repeat this process 20 times. For "Protein" we perform five replications and for "Year" we evaluate on a single split. We compare against VMG with 10 inducing points. Figure 11 summarizes our predictive results and Figure 19 in Appendix C.2 illustrates the shrinkage towards zero exhibited by the recovered solutions. We observe that while HS-BNN is competitive with VMG on the larger data sets, it is outperformed on the smaller ones. In the next section, we describe how the additional regularization of the regularized horseshoe BNN prior provides uniformly better performance.

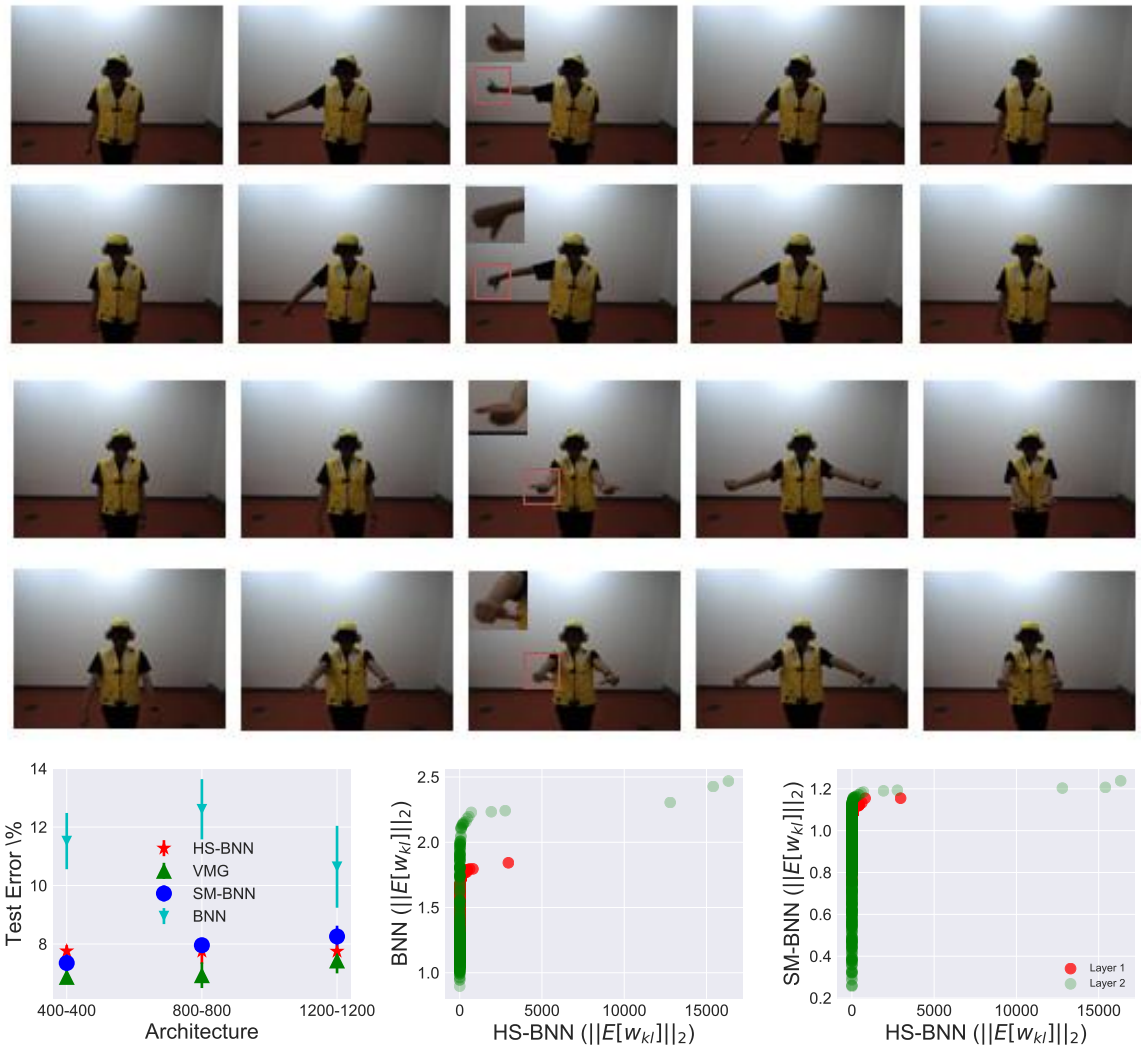


Figure 10: **Gesture recognition**. Top: Example gestures from the NATOPS data set. Bottom: *Left*: Test error rates averaged over 5 runs achieved by competing methods. *Right*: Scatter plots of solutions recovered by HS-BNN and competing methods for the 800-800 architecture on one of the five splits.

### 6.4. Model Innovation: Regularized horseshoe Priors provide consistent benefits, especially on smaller data sets.

Above, we noted that BNNs learned with standard horseshoe priors (HS) can have higher variance with smaller data sets, resulting in lower predictive performance. The regularized Horsehoe prior (reg-HS) should alleviate this issue. We applied the regularized horseshoe prior BNN to the same collection of UCI data sets, with inverse Gamma hyper-parameters

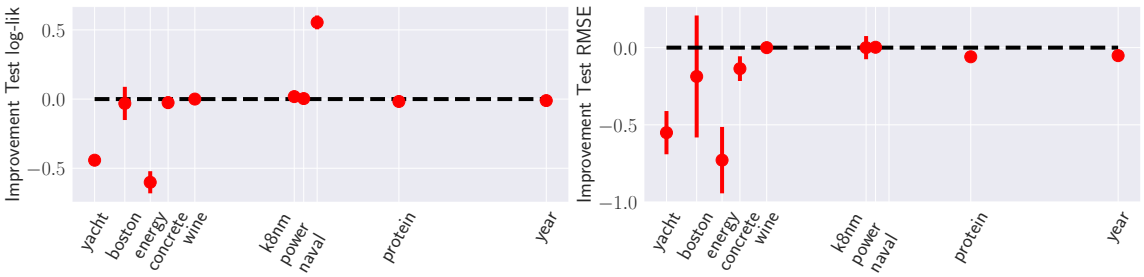


Figure 11: **Regression performance** on a collection of data sets from the UCI repository. Data sets are arranged from left to right in order of increasing number of data points and plotted on the log scale. Plots display relative improvements in test log likelihood and root mean squared error ($\pm$ 1 standard deviation) averaged over 20 random splits of the data. Relative improvement is defined as $(x - y)/max(|x|, |y|)$. VMG performs better on smaller data sets and comparably on larger ones.

$c_a = 2$ and $c_b = 6$ for both the regularized and standard HS-BNN. To focus on the effect of the model, we employed the same factorized variational approximating family for both models. Figure 12 shows that the regularized horseshoe leads to consistent improvements in predictive performance. As expected, the gains are more prominent for the smaller data sets for which the regularization afforded by the regularized horseshoe helps avoid over-fitting.

### 6.5. Inference Innovations: Value of Various Variational Approximations

All of our previous results focused on using fully-factorized approximating families. However, as we noted in Section 4.1, there are many options for adding structure as well as reducing the number of parameters. These choices can have a large effect: in our previous results with comparisons with standard BNNs, the type of approximating family—e.g. the MVG vs. the fully-factored—had a very large effect on performance. In this section, we explore the properties of different variational approximations. All experiments used a batch size of 128, and $b_g = 10^{-5}$.

*Tied approximation speeds up training.* We begin by first comparing the tied and fully factorized approximations on the NATOPS gesture recognition data set, with both models employing un-regularized horseshoe priors. Figure 13 contrasts the held-out test performance of the full and tied approximations against wall-clock time.[1] Since we have far fewer variational parameters to learn, the tied approximation converges faster nearly halving the training time with only small deterioration in predictive performance for classification problems; thus one could also imagine using the tied approximation initially and then switching over to one with more parameters.

*Structured variational approximations provide stronger shrinkage.* In preliminary experiments, we found that of the various approximations described in Section 4.1, the structured approximation outperformed the semi-structured variant. Thus, we focus on a presenting a

1. Experiments were performed on a 2.5 GHz Intel Core i7, with 16GB of RAM.

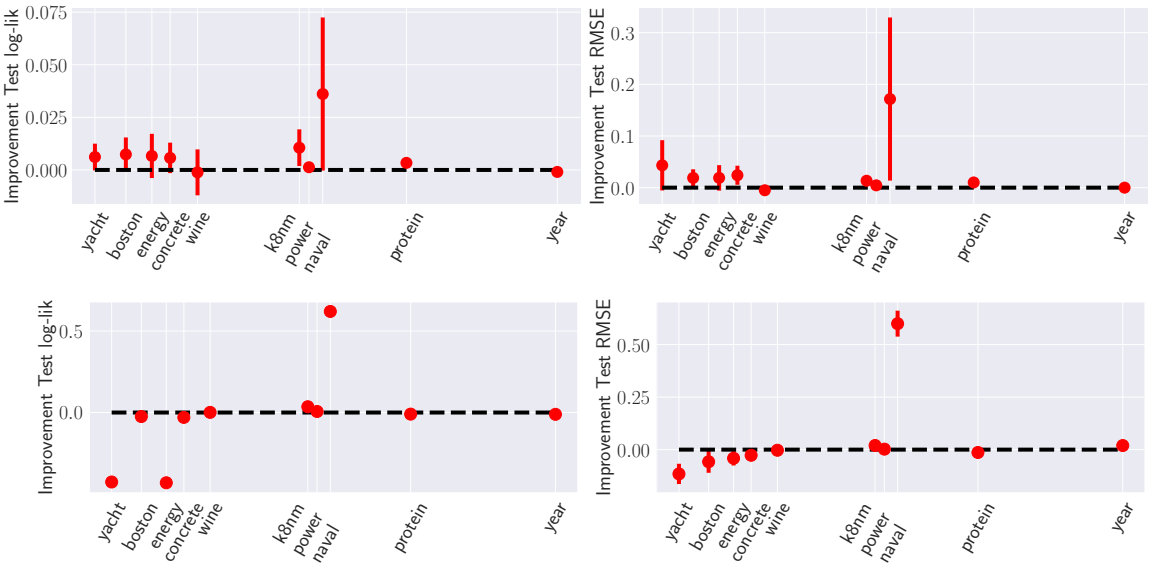


Figure 12: **Regularized horseshoe** results in consistent improvements (TOP) over the vanilla horseshoe prior (values over 0 indicate improvements). The data sets are sorted according to the number of data instances and plotted on the log scale, with 'yacht' being the smallest and 'year' being the largest. Relative improvement is defined as $(x-y)/max(|x|,|y|)$. (BOTTOM) Regularized horseshoe performs similarly to VMG, better on some data sets and worse on others.

detailed comparison between models employing the structured and (fully) factorized variational families. Furthermore, since we have already seen reg-HS outperform vanilla HS, we use regularized horseshoe priors in all subsequent experiments.

*Shrinkage on Synthetic Examples.* In Figures 14 and 15, we explore the effects of structured and factorized variational approximations on predictive uncertainties. We return to the sinusoid prediction problem from Figure 1. We compare single layer 1000 unit BNNs with regularized horseshoe priors employing factorized and structured variational approximations. The structured approximation best alleviates the under-fitting issues and leads to improved fits. We also revisit the noisy polynomial regression example from Secion 6.2. In Figure 15, we find that consistent with the noisy sinusoid example structured approximation results in tighter uncertainties. Consequently, predictive likelihoods do not degrade as more units are added (right panel).

*Shrinkage on UCI benchmarks.* We return to the UCI benchmark to carefully vet the different variational approximations. We deviate from prior work, by using networks with significantly more capacity than previously considered for this benchmark. In particular, we use single layer networks with an order of magnitude more hidden units (500) than considered in previous work (50). This additional capacity is more than that needed to explain the UCI benchmark data sets well. With this experimental setup, we are able to

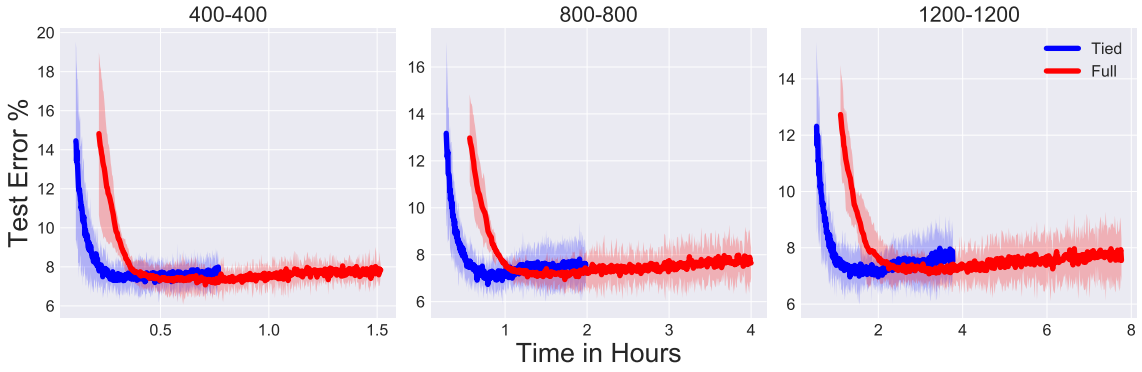


Figure 13: **Tied variational approximation.** We contrast the test accuracy against training time for the tied and full approximations, averaged over five random splits, and run for 500 epochs each on the NATOPS gesture recognition data set.

evaluate how well the proposed methods perform at pruning away extra modeling capacity. For all but the 'year' data set, we report results from 5 trials each trained on a random 90/10 split of the data. For the large year data set, we ran a single trial.

In Figure 16, we see that the factorized and structured variational approximations have similar predictive performance. However, the structured approximation consistently recovers solutions that exhibit much stronger shrinkage towards zero. We have plotted the 50 units with the smallest $||w_{kl}||_2$ weight norms recovered by the factorized and structured approximations, from five random trials. Both approximations provide shrinkage towards zero, but the structured approximation has significantly stronger shrinkage. Further, the degree of shrinkage from the factorized approximation varies significantly between random initializations. In contrast, the structured approximation *consistently* provides strong shrinkage. We compare the shrinkage provided by the two approximations using $||\mathbb{E}[w_{kl}]||_2$ instead of applying the pruning rule from section 4.3 and comparing the resulting compression rates. This is because although the scales $\tau_{kl}\upsilon_l$ inferred by the factorized approximation provide a clear separation between signal and noise, they do not exhibit shrinkage toward zero. On the other hand, $w_{kl} = \tau_{kl}\upsilon_l\beta_{kl}$ does exhibit shrinkage and provides a fair comparison.

*Prediction and shrinkage comparison against competing methods.* Finally, we compare the reg-HS model with structured variational approximation against the variational matrix Gaussian (VMG) approach of Louizos and Welling (2016), which has previously been shown to outperform other variational approaches to learning BNNs. We used the pruning rule with $\delta = 10^{-3}$ for all but the 'year' data set, for which we set $\delta = 10^{-5}$. Figure 16 demonstrates that structured reg-HS is competitive with VMG in terms of predictive performance. We either perform similarly or better than VMG on the majority of the data sets. More interestingly, structured reg-HS achieves competitive performance while pruning away excess capacity and achieving significant compression. We also experimented with fine-tuning the pruned model by updating the weight means while holding others fixed. Figure 16 reports results from running 20 such fine-tuning epochs. We did not find fine-tuning to

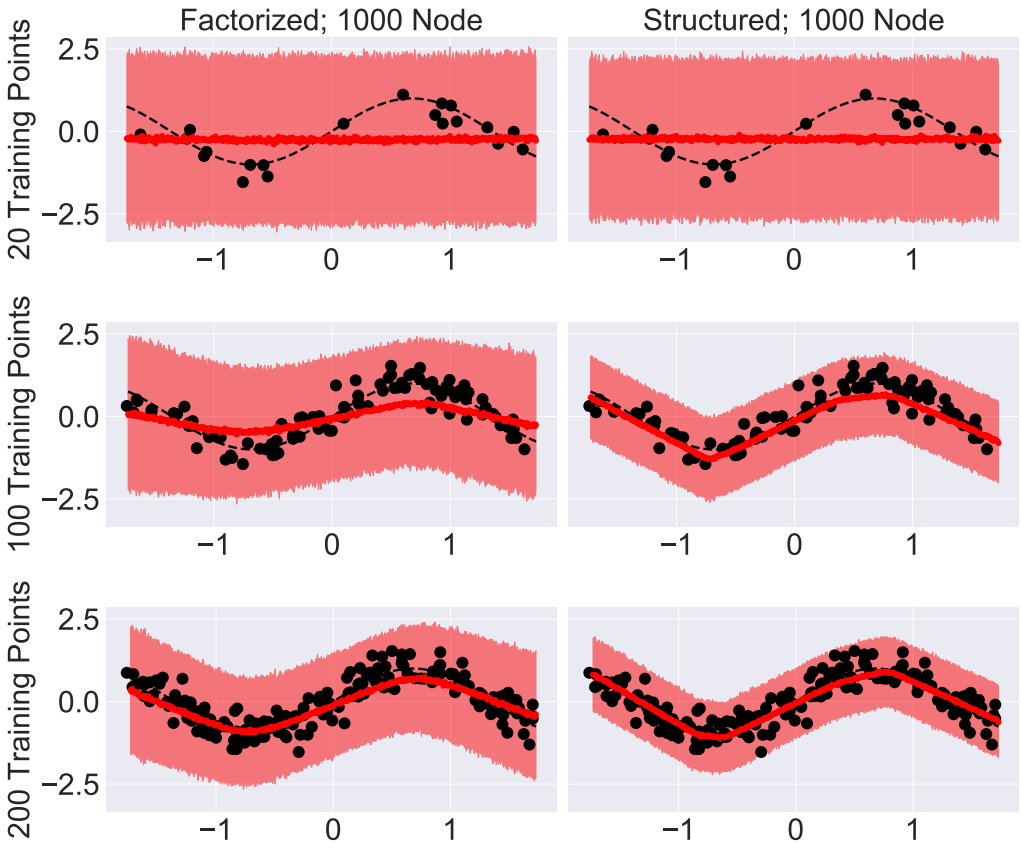


Figure 14: **Regularized horseshoe BNNs** prune away excess capacity and are more resistant to underfitting. Variational approximations aware of model structure improve fits.

significantly improve generalization performance. Additional experiments employing multi-layer architectures are available in the appendix (Figure 23).

### 6.6. Assessing Calibration: HS-BNN with structured variational approximations are better calibrated

The consistency between the predictive distribution and the empirical frequency of observations is referred to as calibration. When the two match well, the predictive model is deemed well-calibrated. Here, we assess calibration of BNN variants on the synthetic data presented in Figures 1, 5, and 15. Knowing the true data generating process, we sample an additional thousand data points unobserved during training. We use this validation set to assess model calibration and compare a BNN with $\mathcal{N}(0, 1)$ priors, against the regularized HS-BNN employing both the structured and factorized variational approximations. Since calibration is only a meaningful metric provided that the fits are good, we assess performance of the

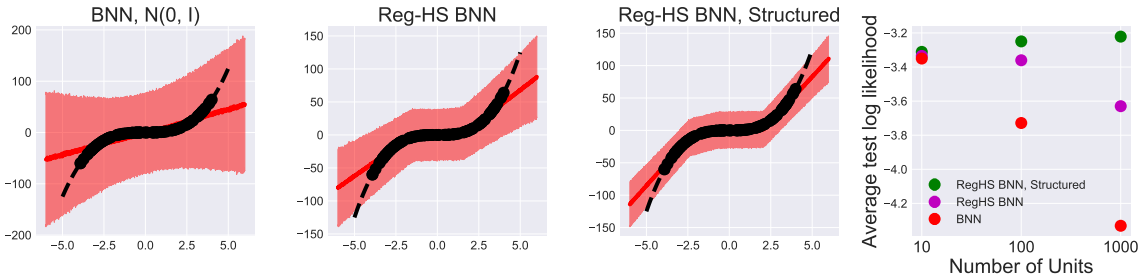


Figure 15: **Structured variational approximations** improve model fits. The crosses indicate training data and the dotted line visualizes the noise free cubic function, and in red we visualize the predicted means and ± 3 standard deviations for single layer networks with 1000 hidden units. Right: Average predictive log likelihood for single layer networks with 10, 100 and 1000 units. Regularized HS-BNN with structured variational approximation provides robust performance in the presence of model misspecification.

models trained on two hundred data points. With these many training instance, all three models achieve a reasonable fit to the data.

Calibrations of continuous responses are typically assessed using the predictor's predictive cumulative distribution functions (CDF). Before presenting our results, we briefly review the definitions of the two types of calibration we consider here and refer the interested reader to (Gneiting et al., 2007) for a more thorough treatment of the material.

Let $n$ index the data $(\{x_n, y_n\}_{n=1}^N)$ in the validation set, $F_n$ be the predictive CDF, i.e., CDF of the predictive distribution $p(y_n \mid x_n) = \int p(y_n \mid x_n, \mathcal{W}) p(\mathcal{W} \mid \mathcal{D}_{train}) d\mathcal{W}$. In our experiments we used the empirical distribution function of Monte Carlo samples to approximate $F_n$.[2] Let $G_n$ be the (unobserved) true data generating process. We say that $\{p_n = F_n(y_n)\}_{n=1}^N$ is probabilistically calibrated with respect $\{G_n\}_{n=1}^N$, iff $\frac{1}{N}\sum_{n=1}^N \mathbf{1}(p_n \leq p)$ converges almost surely to $p$ for all $p \in [0, 1]$. The definition can be operationalized by selecting a sufficiently large $N$, a set of discrete thresholds $(p_t)$ in $[0, 1]$ and computing the fraction of validation instances below the threshold and comparing it to the threshold itself. This is illustrated in Figure 17, where we plot the thresholds $p_t$ (expected confidence level) against $\hat{p}_t = \frac{1}{N}\sum_{n=1}^N \mathbf{1}(p_n \leq p_t)$ (empirical confidence level). Plots closer to the diagonal indicate better calibration. Probabilistic calibration has also been recently used to assess the calibration of deep regression models (Kuleshov et al., 2018). They also proposed a quantitative metric for measuring miscalibration, $\propto \sum_{t=1}^T (p_t - \hat{p}_t)^2$. Although widely used probabilistic calibration is a necessary but not sufficient criteria for calibration (Gneiting et al., 2007; Hamill, 2001). We assess the models according to an additional mode of calibration — marginal calibration, to further lend credence to whether a particular model is indeed well calibrated.

Marginal calibration is said to hold iff the empirical distribution function of the validation set, $\hat{G}(y) = \frac{1}{N}\sum_{n=1}^N \mathbf{1}(y_n < y)$, $y \in \mathbb{R}$ converges almost surely to the average

2. A moment matched Gaussian CDF produced similar results.

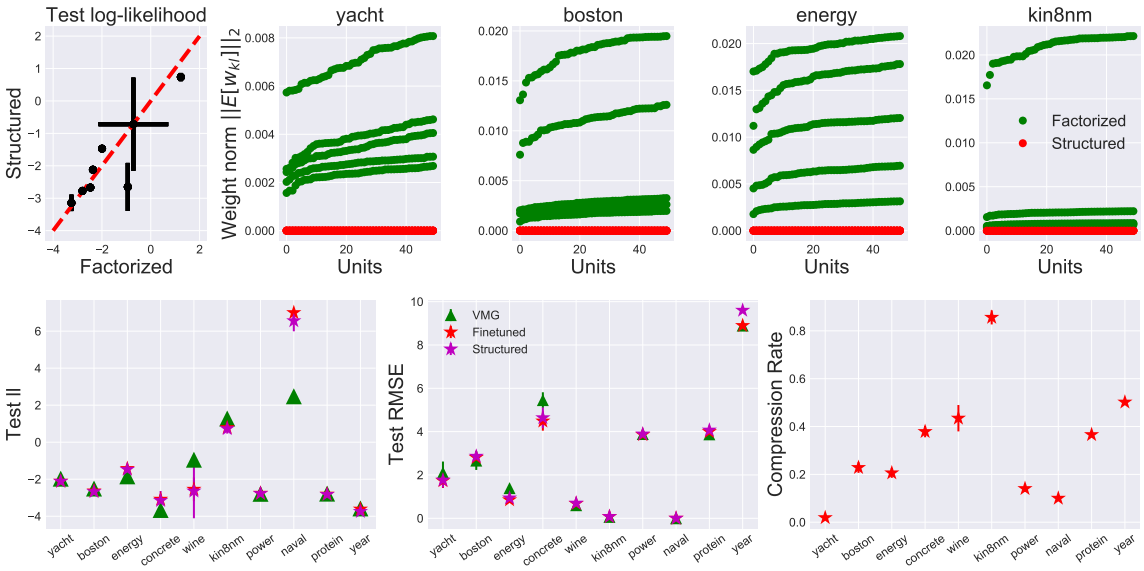


Figure 16: *Top*: **Structured variational approximations** result in similar predictive performance but consistently recover solutions that exhibit stronger shrinkage. The left most figure plots the predictive log likelihoods achieved by the two approximations, each point corresponds to a UCI data set. We also plot the fifty units with the smallest $||E[w_{kl}]||_2$, on a number of data sets. Each point in the plot displays the inferred $||E[w_{kl}]||_2$ for a unit in the network. We plot recovered expected weight norms from all five random trials for both the factorized and structured approximation. The structured approximation (in red) consistently provides stronger shrinkage. The factorized approximation both produces weaker shrinkage and the degree of shrinkage exhibits higher variance with random trials. *Bottom*: The structured approximation is competitive with VMG while using much smaller networks. Fine tuning leads to small improvements. Compression rates are defined as the fraction of un-pruned units.

predictive CDF $\bar{F}(y) = \lim_{N\to\infty} \frac{1}{N}\sum_{n=1}^{N} F_n(y)$ for all $y \in \mathbb{R}$ as $N \to \infty$. The definition can be approximated by selecting a sufficiently large validation set and comparing the empirical distribution function computed from the validation set and $\bar{F}$ both evaluated at a common set of selected thresholds $y_t$. The empirical distribution function and average predictive CDF are compared in Figure 17. Plots closer to the diagonal again indicate better calibration. We can again quantify the calibration error by computing the error $\propto \sum_{t=1}^{T}(\hat{G}(y_t) - \bar{F}(y_t))^2$. We define overall calibration error as the arithmetic mean,

$$\text{calibration error} = \frac{1}{2}\left(\frac{1}{T_1}\sum_{t=1}^{T_1}(p_t - \hat{p}_t)^2 + \frac{1}{T_2}\sum_{t=1}^{T_2}(\hat{G}(y_t) - \bar{F}(y_t))^2\right)$$

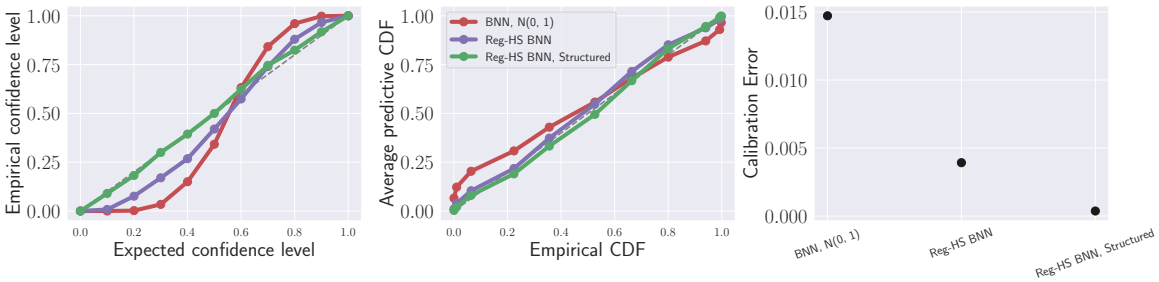


Figure 17: **Calibration assessments**. *Left to Right*: Probabilistic calibration, Marginal calibration, and calibration error. For the first two, plots closer to the diagonal indicate better calibration.

### 6.7. HS-BNNs Application: Reinforcement Learning

One application area in which having good predictive uncertainty estimates is crucial is in model-based reinforcement learning scenarios (e.g. (Depeweg et al., 2017; Gal et al., 2016b; Killian et al., 2017)): here, it is essential not only to have an estimate of what state an agent may be in after taking a particular action, but also an accurate sense of all the states the agent may end up in. In the following, we apply our regularized HS-BNN with structured approximations to two domains: the 2D map of Killian et al. (2017) and acrobot Sutton and Barto (1998). The goal of the 2D map was to traverse from a start region to a goal region around an obstacle and "wind" or slippery conditions. The state space ($S \in R^2$) of 2D map consists of $x$-axis and $y$-axis and at each step, the agent can choose actions from directions $N, E, S, W$.

The goal of the acrobot was to swing up an under-actuated double pendulum where the goal is to swing it from hanging down to fully up. The state space ($S \in R^4$) consists of two angles (the angle between the pendulum and the ceil and the angle of the joint), the angular velocities of two links and there are three possible actions. The details of the dynamics settings of both domains are described in (Killian et al., 2017).

In each domain, we first collected data by training a DDQN (2 layers of width 256, 512; learning rate $5e-4$) online (updated every episode) and an epsilon-greedy policy that started at 1 and decayed to 0.15 with decay rate 0.995. That is, we started with a randomly initialized DDQN, ran it with a completely random policy (epsilon=1), updated the parameters of the DDQN to estimate the state-action values based on the data. We repeated the process over 500 episodes, gradually decaying epsilon to have the agent converge to the optimal policy. (2 layers of width 256, 512; learning rate $5e-4$). Overall, this procedure ensured that we had a wide variety of samples that were still biased in coverage toward the optimal policy.

While this procedure allowed us to generate a large amount of data around the optimal policy, our goal was learning in resource-constrained settings. Thus, we sampled 1% data randomly from the large cohort. We used these data to train a transition function to predict the next position of the agent given the current position and action. Using a BNN to model the transition function allowed us to capture uncertainty due to limited data, as well as

Table 2: Model-based reinforcement learning. The under-fitting of the standard BNN results in lower task performance, whereas the HS-BNN is more robust to this underfitting.

| | 2D Map | |
|---|---|---|
| | Test RMSE | Avg. Reward |
| BNN x-500-y | 0.187 | 975.386 |
| BNN x-100-100-y | 0.089 | 966.716 |
| **Structured x-500-y** | **0.058** | **995.416** |
| Structured x-100-100-y | 0.061 | 992.893 |
| | Acrobot | |
| BNN x-500-y | 0.924 | -156.573 |
| BNN x-100-100-y | 0.710 | -23.419 |
| Structured x-500-y | **0.558** | -108.443 |
| **Structured x-100-100-y** | 0.656 | **-17.530** |

stochasticity. For the transition function, we considered two architectures, a single hidden layer network with 500 units, and a two layer network with 100 units per layer as the transition function for each domain. We used learning rate 0.0002, and batch size 32, and trained the network for 2000 episodes. We considered two priors, the standard normal on the weights and our HS-BNN.

Finally, once we had trained an approximate transition function to model the domain, we used that function as a simulator. Now, instead of training a DDQN based on data from the real environment, we trained with data generated from the BNN. Once that policy learning converged, we tested that policy—learned from the approximate BNN transition model—on the original simulator to measure its performance in the true setting.

*HS-BNNs improve reinforcement learning performance.* As in our prediction results, training a moderately-sized BNN with so few data results in severe underfitting, which in turn, adversely affects the quality of the policy that is learned. That is, if we believe we have a lot of uncertainty about the effect of an action—due to underfitting—we cannot find an appropriate policy. We see in table 2 that the better fitting of the structured reg-HS-BNN results in higher task performance, across domains and model architectures.

## 7. Discussion and Conclusion

Model selection in Bayesian networks is an important problem. In this work, we demonstrated that a properly parameterized horseshoe prior on the scales of the weights incident to each node is a computationally efficient tool for model selection in Bayesian neural networks. Decomposing the horseshoe prior into inverse gamma distributions led to computationally convenient inference algorithms. However, auxiliary variables introduced by the decomposition can pose further inferential challenges and lead to poorer approximations (Neville et al., 2014). We found that using a non-centered representation ensured a degree of robustness to such issues as well as poor local optima. Designing efficient and ac-

curate inference algorithms as well as metrics for reliably measuring the quality of posterior approximations for BNNs and Horseshoe BNNs remains an exciting direction for followup work. We also showed that the regularized horseshoe prior, combined with a structured variational approximating distribution, provides a computationally efficient tool for model selection in Bayesian neural networks. By retaining crucial posterior dependencies, the structured approximation provided strong shrinkage while being competitive in predictive performance to approaches without shrinkage.

All that said, one might wonder about alternatives for model selection. For example, one could place a simple exponential prior on the weight scale to encourage all incident weights to be zero, but without heavy tails all scales will be forced to be artificially low and prediction will suffer (Molchanov et al., 2017; Wen et al., 2016). In contrast, simply using a heavy-tail prior on the scale parameter, such as a half-Cauchy, will not apply any pressure toward setting small scales to zero. Another alternative is to observe that a node can be pruned if the product $z \cdot w$ is nearly constant for all inputs $z$—having small weights is sufficient to achieve this property; weights $w$ that are orthogonal to the variation in $z$ is another. Thus, instead of putting a prior over the scale of $w$, one could put a prior over the scale of the variation in $z \cdot w$. However, in initial experiments, we found that such a formulation has many more local optima and thus harder to optimize. In contrast, inference for the horseshoe prior is fairly robust: following the parameterization considerations and choice of variational family, our code is standard BBVI.

Within Horseshoe priors, there remain several interesting follow-on directions, including, modeling enhancements that use layer, node, or even weight specific weight decay $c$, or layer specific global shrinkage parameter $b_g$ to provide different levels of shrinkage to different parts of the BNN. One could imagine using the machinery of Horseshoe priors to turn off entire layers or entire skip connections; one could imagine these being governed in some way by the input location $x$. It would also be interesting to consider how to adapt Horseshoe priors to exhibit interesting limiting behaviors as the number of nodes in a layer grow without bound.

More broadly, model selection via Horseshoe priors may represent a sweet spot, at least for now, between (still the most common) fully factorized priors and recent attempts at specifying function-based priors (Wang et al., 2019; Sun et al., 2019). The latter provide an elegant approach for specifying an implicit prior over BNN weights via an explicit prior over the kinds of functions are likely, but currently these approaches are fairly computationally intensive and involve more challenging optimizations. It would be interesting to see if variants of Horseshoe priors can bring us interesting function classes at relatively low computational and inferential expense.

## Acknowledgments

JY acknowledges support from NSF RI-1718306. JY and FDV acknowledge support from the IBM Faculty Award. We would also like to thank the anonymous reviewers for their excellent suggestions, Karen Ullrich and Christos Louizos for their comments on an early version of the manuscript.

## Appendix A. Algorithm

### A.1. Conditional variational pre-activations

Recall from Section 4.2, that the variational pre-activation distribution is given by $q(b \mid a, \nu_l, \phi_{\beta_l}) = \mathcal{N}(b \mid \mu_b, \Sigma_b) = \mathcal{N}(b \mid M_{\beta_l|\nu_l}^T a, (a^T U_{\beta_l|\nu_l} a)V)$, where $U = \Psi + hh'$, and $V$ is diagonal. The Equation requires $M_{\beta_l|\nu_l}$ and $U_{\beta_l|\nu_l}$. The expressions for these follow directly from the properties of partitioned Gaussians.

For a particular layer $l$, we drop the explicit dependency on $l$ from the notation. Recall that $B = \begin{bmatrix} \beta \\ \nu^T \end{bmatrix}$, and let $B \in \mathbb{R}^{m \times n}$, $\beta \in \mathbb{R}^{m-1 \times n}$, and $\nu \in \mathbb{R}^{n \times 1}$ $q(B \mid \phi_B) = \mathcal{MN}(B \mid M, U, V)$. From properties of the Matrix normal distribution, we know that a column-wise vectorization of $B$, $\vec{B} \sim \mathcal{N}(\vec{M}, V \otimes U)$. From this and Gaussian marginalization properties it follows that the $j^{\text{th}}$ column $t_j = [\beta_j; \nu_j]$ of $B$ is distributed as $t_j \sim \mathcal{N}(m_j, V_{jj}U)$, where $m_j$ is the appropriate column of $M$. Conditioning on $\nu_j$ then yields, $q(\beta_j \mid \nu_j) = \mathcal{N}(\beta_j \mid \mu_{\beta_j|\nu_j}, \Sigma_{\beta_j|\nu_j})$, where

$$\Sigma_{\beta_j|\nu_j} = V_{jj}(\Psi_\beta + \frac{\Psi_\nu}{\Psi_\nu + h_\nu^2} h_\beta h_\beta^T)$$
$$\mu_{\beta_j|\nu_j} = \mu_{\beta_j} + \frac{h_\nu(\nu_j - \mu_{\nu_j})}{\Psi_\nu + h_\nu^2} h_\beta$$

Rearranging, we can see that, $M_{\beta|\nu}$ is made up of the columns $\mu_{\beta_j|\nu_j}$ and $U_{\beta|\nu} = \Psi_\beta + \frac{\Psi_\nu}{\Psi_\nu + h_\nu^2} h_\beta h_\beta^T$.

### A.2. Algorithmic details

The ELBO corresponding to the non-centered regularized HS model is,

$$\begin{aligned}
\mathcal{L}(\phi) = {} & \mathbb{E}_{q(c^2)}[\ln \text{ Inv-Gamma}(c^2 \mid c_a, c_b)] + \mathbb{E}_{q(\kappa^2)q(\rho_k)}[\ln \text{ Inv-Gamma}(\kappa^2 \mid 1/2, 1/\rho_\kappa)] \\
& + \mathbb{E}_{q(\rho_k)}[\ln \text{ Inv-Gamma}(\rho_\kappa \mid 1/2, 1/b_\kappa^2)] + \sum_n \mathbb{E}_{q(\beta, \mathcal{T}, \kappa^2, c^2)}[\ln\, p(y_n \mid \beta, \mathcal{T}, \kappa^2, c^2, x_n)] \\
& + \sum_{l=1}^{L-1} \sum_{k=1}^{K_L} \mathbb{E}_{q(\lambda_{kl})}[\ln \text{ Inv-Gamma}(\lambda_{kl} \mid 1/2, 1/b_0^2)] \\
& + \sum_{l=1}^{L-1} \mathbb{E}_{q(\upsilon_l)q(\vartheta_l)}[\ln \text{ Inv-Gamma}(\upsilon_l^2 \mid 1/2, 1/\vartheta_l)] + \mathbb{E}_{q(\vartheta_l)}[\ln \text{ Inv-Gamma}(\vartheta_l \mid 1/2, 1/b_g^2)] \\
& + \sum_{l=1}^{L-1} \mathbb{E}_{q(B_l)}[\ln \mathcal{N}(\beta_l \mid 0, \mathbb{I})] + \sum_{l=1}^{L-1} \sum_{k=1}^{K_L} \mathbb{E}_{q(B_l)}[\ln \text{ Inv-Gamma}(\tau_{kl}^2 \mid 1/2, 1/\lambda_{kl})] \\
& + \sum_{k=1}^{K_L} \mathbb{E}_{q(\beta_{kL})}[\ln \mathcal{N}(\beta_{kL} \mid 0, \mathbb{I})] + \mathbb{H}[q(\theta \mid \phi)].
\end{aligned}$$

We rely on a Monte-Carlo estimates to evaluate the expectation involving the likelihood $\mathbb{E}_{q(\beta, \mathcal{T}, \kappa^2, c^2)}[\ln\, p(y_n \mid \beta, \mathcal{T}, \kappa^2, c^2, x_n)]$.

**Efficient computation of the Matrix Normal Entropy** The entropy of $q(B) = \mathcal{MN}(B \mid M, U, V)$ is given by $\frac{mn}{2}\ln(2\pi e) + \frac{1}{2}\ln|V \otimes U|$. We can exploit the structure of $U$ and $V$ to compute this efficiently. We note that $\ln|V \otimes U| = m\ln|V| + n\ln|U|$. Since $V$ is diagonal $\ln|V| = \sum_j \ln V_{jj}$. Using the matrix determinant lemma we can efficiently compute $|U| = (1 + h'\Psi^{-1}h)|\Psi|$. Owing to the diagonal structure of $\Psi$, computing it's determinant and inverse is particularly efficient.

**Fixed point updates** The auxiliary variables $\rho_\kappa$, $\vartheta_l$ and $\vartheta_l$ all follow inverse Gamma distributions. Here we derive for $\lambda_{kl}$, the others follow analogously. Consider,

$$\begin{aligned}\ln q(\lambda_{kl}) &\propto \mathbb{E}_{-q_{\lambda_{kl}}}[\ln \text{Inv-Gamma}(\tau_{kl}^2 \mid 1/2, 1/\lambda_{kl})] + \mathbb{E}_{-q_{\lambda_{kl}}}[\ln \text{Inv-Gamma}(\lambda_{kl} \mid 1/2, 1/b_0^2)], \\ &\propto (-1/2 - 1/2 - 1)\ln \lambda_{kl} - (\mathbb{E}[1/\tau_{kl}] + 1/b_0^2)(1/\lambda_{kl}),\end{aligned} \tag{10}$$

from which we see that,

$$\begin{aligned}q(\lambda_{kl}) &= \text{Inv-Gamma}(\lambda_{kl} \mid c, d), \\ c = 1, d &= \mathbb{E}[\frac{1}{\tau_{kl}^2}] + \frac{1}{b_0^2}.\end{aligned} \tag{11}$$

Since, $q(\tau_{kl}^2) = \ln \mathcal{N}(\mu_{\tau_{kl}}, \sigma_{\tau_{kl}}^2)$, it follows that $\mathbb{E}[\frac{1}{\tau_{kl}^2}] = \exp\{-\mu_{\tau_{kl}} + 0.5 * \sigma_{\tau_{kl}}^2\}$. We can thus calculate the necessary fixed point updates for $\lambda_{kl}$ conditioned on $\mu_{\tau_{kl}}$ and $\sigma_{\tau_{kl}}^2$. Our algorithm uses these fixed point updates given estimates of $\mu_{\tau_{kl}}$ and $\sigma_{\tau_{kl}}^2$ after each Adam step.

## Appendix B. Useful Properties of the Log-Normal distribution

These results are well known and included here for completeness. See Johnson et al. (1970), or other appropriate text for more details. If $\ln x \sim \mathcal{N}(\mu, \sigma^2)$ then $x$ follows a log-Normal distribution with mean, $\mathbb{E}[x] = e^{\mu + \frac{1}{2}\sigma^2}$ and variance, $\mathbb{E}[x^2] - \mathbb{E}[x]^2 = (e^{\sigma^2} - 1)e^{2\mu + \sigma^2}$.

### B.1. Moments

For any real $k$, the $k^{\text{th}}$ moment of $x$ is,

$$\mathbb{E}[x^k] = e^{k\mu + \frac{1}{2}k^2\sigma^2},$$

this immediately leads to the result $\mathbb{E}[x^{-1}] = e^{-\mu + \frac{1}{2}\sigma^2}$, which we used to compute the parameters of the inverse Gamma distribution in Equation 11.

### B.2. Closed under multiplication

If $x_a \sim \text{log-Normal}(\mu_a, \sigma_a^2)$ and $x_b \sim \text{log-Normal}(\mu_b, \sigma_b^2)$, then,

$$y = x_a x_b \sim \text{log-Normal}(\mu_a + \mu_b, \sigma_a^2 + \sigma_b^2),$$

a fact we exploited for computing with the pruning rule of Section 4.3.

## Appendix C. Experiments

### C.1. Experimental details

For regression problems we use Gaussian likelihoods with an unknown precision $\gamma$, $p(y_n \mid f(\mathcal{W}, x_n), \gamma) = \mathcal{N}(y_n \mid f(\mathcal{W}, x_n), \gamma^{-1})$. We place a vague prior on the precision,$\gamma \sim$ Gamma$(6, 6)$ and approximate the posterior over $\gamma$ using another variational distribution $q(\gamma \mid \phi_\gamma)$. The corresponding variational parameters are learned via a gradient update during learning.

**Classification Experiments** For VMG, we used 150 inducing points, a batch size of 128, and a learning rate of 0.001.

**Regression Experiments** For comparing the reg-HS and HS models we followed the protocol of (Hernandez-Lobato & Adams, 2015) and trained a single hidden layer network with 50 rectified linear units for all but the larger "Protein" and "Year" data sets for which we train a 100 unit network. For the smaller data sets we train on a randomly subsampled 90% subset and evaluate on the remainder and repeat this process 20 times. For "Protein" we perform 5 replications and for "Year" we evaluate on a single split. For, VMG we used 10 pseudo-inputs, a learning rate of 0.001 and a batch size of 128.

The details of the different UCI data sets used are presented in Table 3

| data set | N | D |
|---|---|---|
| Yacht | 308 | 6 |
| Boston | 506 | 13 |
| Energy | 768 | 8 |
| Concrete | 1030 | 8 |
| Wine | 1599 | 11 |
| Kin8nm | 8192 | 8 |
| Power Plant | 9568 | 4 |
| Naval | 11,934 | 16 |
| Protein | 45,730 | 9 |
| Year | 515,345 | 90 |

Table 3: UCI Regression results. HS-BNN and VMG are compared.

### C.2. Additional Experiments

#### C.2.1. Non-centered parameterization

In Figure 18 we display that the non-centered parameterization is able to recover the true network even under large model misspecification.

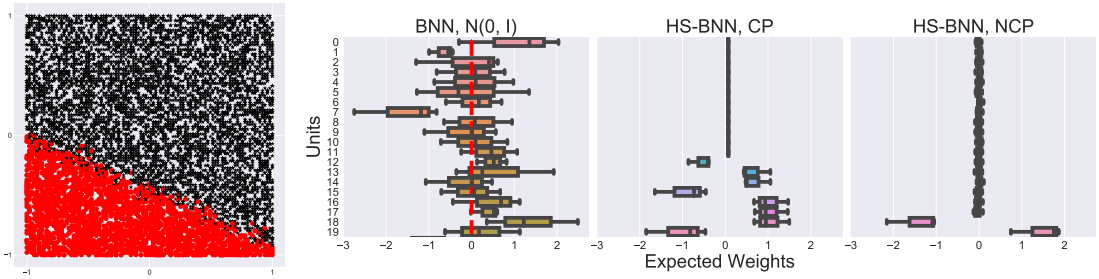


Figure 18: **Non-centered parameterization** is essential for robust inference. From left to right, A synthetic nearly linear classification problem generated from sampling a 2-2-1 network, the two classes are in red and black. Expected weights inferred with a 100 unit Bayesian neural network with Gaussian priors on weights. Expected weights recovered with a 100 unit centered horseshoe parameterization and a 100 unit Non-centered horseshoe parameterization. Only the 20 units with the largest weights are displayed.

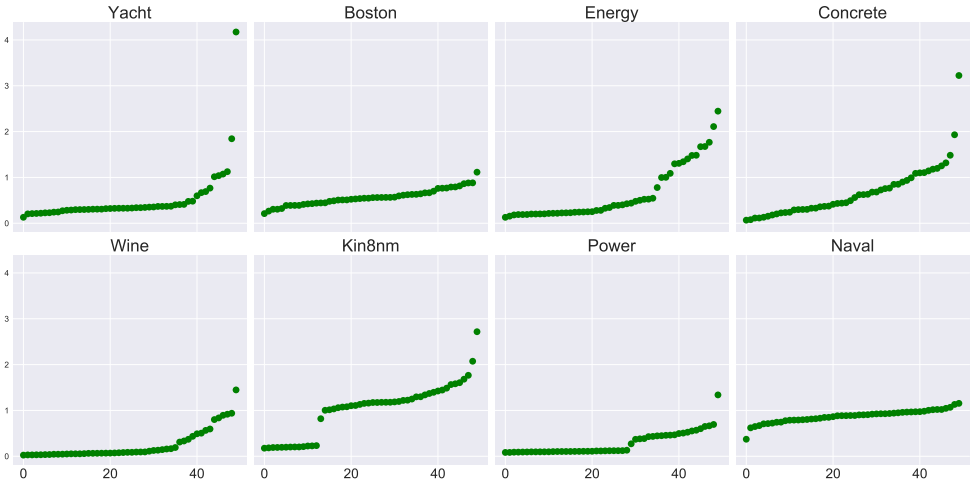


Figure 19: We plot $||w_{kl}||_2$ recovered by the HS-BNN using the fully factorized variational approximation on a number of UCI data sets.

### C.2.2. Shrinkage provided by the fully factorized approximation on UCI benchmarks

Figure 19 illustrates the shrinkage afforded by 50 unit HS-BNNs using fully factorized approximations. Note that on some data sets, we do not achieve much compression and all 50 units are used. This is a consequence of the fully factorized approximations providing weaker shrinkage as well as 50 units not being large enough to model the complexity of the data set.

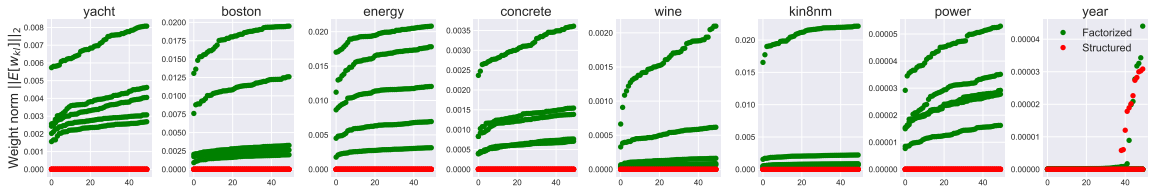


Figure 20: Structured variational approximations consistently recover solutions that exhibit stronger shrinkage. We plot the 50 smallest $||w_{kl}||_2$ recovered by the two approximations on five random trials on a number of UCI data sets.

### C.2.3. Scaled Priors and Structured Inference

Here we further explore under-fitting issues in BNNs *not* employing horseshoe priors. As pointed out in the main text, using $\mathcal{N}(0, \sigma^2)$ priors over the network weights encourages the prior predictive variance to increase with the width of the network. It is well known (Neal, 1997) that instead scaling the prior variance with the width of the layer $\mathcal{N}(0, \sigma^2/H)$ does not result in such pathologies and one instead recovers a Gaussian process in the infinite width limit. A natural question then is whether such a scaled prior alleviates the under-fitting issues observed in Figure 1. We find (Figure 21) that this is not the case when employing a fully factorized variational approximation. We find that the posterior again tends to the prior just as it did for BNNs with standard normal priors. Since, here we are using one dimensional inputs the input to hidden layer prior is still a $\mathcal{N}(0, 1)$ distribution while the hidden to output layer employs a $\mathcal{N}(0, 1/1000)$ distribution. Empirically, we find that the higher variance input to hidden layer compensates for the smaller variance output layer prior and cause high predictive uncertainties. A heuristic, sometimes employed, is to divide the input to each layer by the square root of its dimensionality during the variational forward pass. This addresses the high predictive variance issue but not the underfitting problem, resulting in similar poor fits but with lower predictive variances. These observations suggest that in addition to appropriate priors better variational approximations are necessary to avoid pathologies. We further corroborated this hypothesis by instead using layer-wise matrix variate Gaussians (Louizos and Welling, 2016) as our variational approximation. We observe that even with $\mathcal{N}(0, \sigma^2)$ priors improved fits are obtained with approximations that retain more of the correlations among the weights. As illustrated in the main text similar conclusions also hold for HS-BNN — structured inference leads to higher test likelihoods (Figure 16) and better calibrated uncertainties (Figures 14, 17).

### C.2.4. Comparisons against Hamiltonian Monte Carlo

Retaining model structure in the variational approximations do typically lead to more accurate but nonetheless biased inferences. Hamiltonian Monte Carlo (Neal, 1993; Duane et al., 1987) on the other hand provides accurate and asymptotically exact inferences and is widely regarded as the gold standard for inference in Bayesian neural networks. They do however have difficulty scaling with the number of data instances. Moreover, the horseshoe is a challenging distribution to sample from, these difficulties are only exacerbated for BNN models with (regularized) horseshoe priors which involve sampling high dimensional poste-

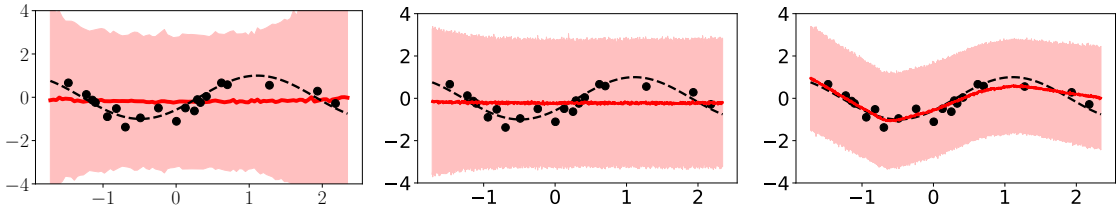

Figure 21: **Model or Inference**. Predictive distributions from a single layer BNN with ReLU activations, containing 1000 units trained on twenty noisy samples from a Sine function. From left to right, we have a BNN with $\mathcal{N}(0, \mathbb{I}/H_{in})$ priors over weights and a fully factorized variational approximation, and BNNs with $\mathcal{N}(0, \mathbb{I})$ priors but employing matrix Gaussian variational distributions (Louizos and Welling, 2016) with one and two pseudo inputs. Similar to Figure 1, underfitting issues can be seen for the scaled priors employing fully factorized variational families. More expressive variational approximations (columns 2 and 3) somewhat alleviate these issues, even for the unscaled $\mathcal{N}(0, \mathbb{I})$ priors.

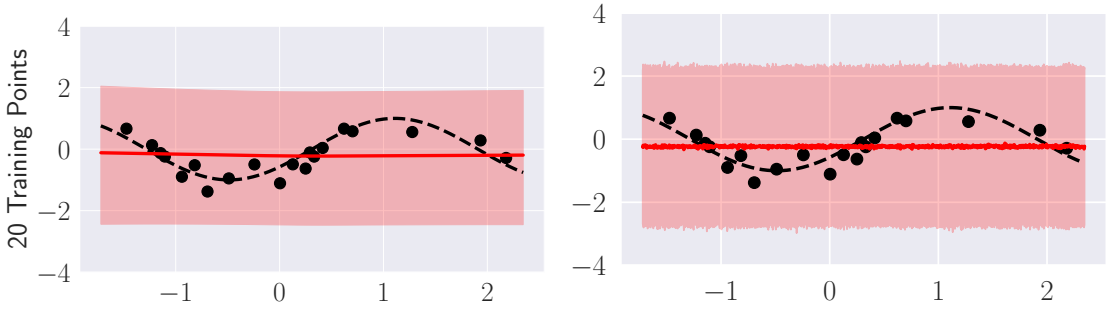


Figure 22: **HMC** (*left*) and **structured VI** (*right*) predictive distributions of a single hidden layer ten unit BNN with a regularized horseshoe prior ($b_g = 10^{-5}$).

riors involving many horseshoe distributions, one for each hidden layer node. Nonetheless, resorting to Riemannian Hamiltonian Monte Carlo (Girolami and Calderhead, 2011) we sampled from the posterior of a small BNN model with a single hidden layer comprising of ten nodes, conditioned on twenty training instances. We used the diagonal *softabs* metric proposed in (Betancourt, 2013). Figure 22 compares the predictive distribution resulting from 1000 samples from HMC against structured VI, with both approaches performing similarly. We emphasize that it is unlikely that structured VI continues to perform similarly to HMC on larger models and larger amounts of data. Developing HMC procedures that reliably sample from larger BNNs employing similar shrinkage priors is an exciting direction of future work.

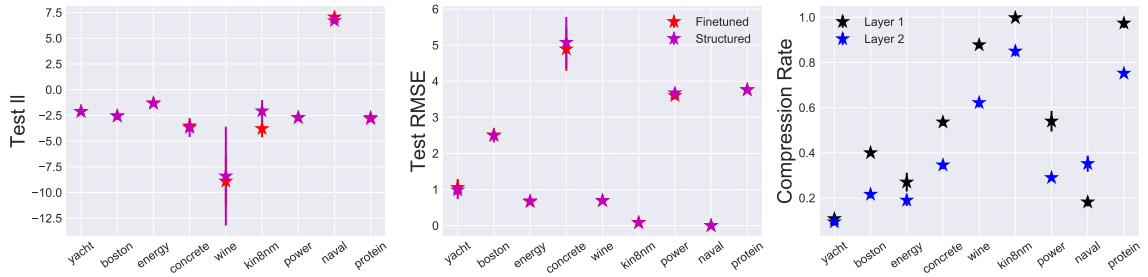


Figure 23: **Structured variational approximations** results for two layer networks each with 100 hidden units. We achieve competitive performance, often improving on the x-500-y networks. For most data sets the second hidden layer is exhibits larger compression. Compression rates are defined as the fraction of un-pruned units.

### C.2.5. Additional UCI Regression results

In Figure 23 we present additional regression results for x-100-100-y networks employing regularized horseshoe priors with structured variational approximations. We report results from averaging over five 90/10 random splits of the data set. We use $b_g = 10^{-5}$ and a pruning threshold $\delta = 10^{-3}$. Observe that in addition to competitive predictive performance, we achieve (often significant) compression for both layers. Interestingly, we find that for most data sets our pruning strategy more severely prunes the second layer (closer to the output).

## Appendix D. Metrics

For a test point $i$, with true label $y_i^*$ and posterior predictive samples $\{y_i^1, \ldots, y_i^S\}$ we compute the per data point root mean squared error as,

$$\text{rmse}_i = \sqrt{\frac{1}{S}\sum_{s=1}^{S}(y_i^s - y_i^*)^2},$$

and the average root mean squared error of the test split $t$ with $N$ test points as,

$$\bar{\text{rmse}}_t = \frac{1}{N}\sum_{i=1}^{N}\text{rmse}_i = \frac{1}{N}\sum_{i=1}^{N}\sqrt{\frac{1}{S}\sum_{s=1}^{S}(y_i^s - y_i^*)^2}.$$

The mean and standard deviation of $\bar{\text{rmse}}_t$ across splits $t = 1, \ldots T$ is reported in the text.

We compute test log likelihood for a test point $i$, $\{y_i^*, x_i^*\}$ given training data $\mathcal{D}$ as,

$$\text{ll}_i = \ln p(y_i^* \mid x_i^*, \mathcal{D}) \approx \ln \frac{1}{S}\sum_{s=1}^{S} p(y_i^* \mid x_i^*, \mathcal{W}^s); \quad \mathcal{W}^s \sim q(\mathcal{W} \mid \mathcal{D}).$$

Summaries across data instances and train/test splits are computed and reported analogously to root mean squared error. For split $t$ with $N$ test instances we have,

$$\bar{\text{ll}}_t = \frac{1}{N}\sum_{i=1}^{N} \text{ll}_i = \frac{1}{N}\sum_{i=1}^{N} \ln \frac{1}{S}\sum_{s=1}^{S} p(y_i^* \mid x_i^*, \mathcal{W}^s); \quad \mathcal{W}^s \sim q(\mathcal{W} \mid \mathcal{D}).$$

## Appendix E. Sparsity inducing priors.

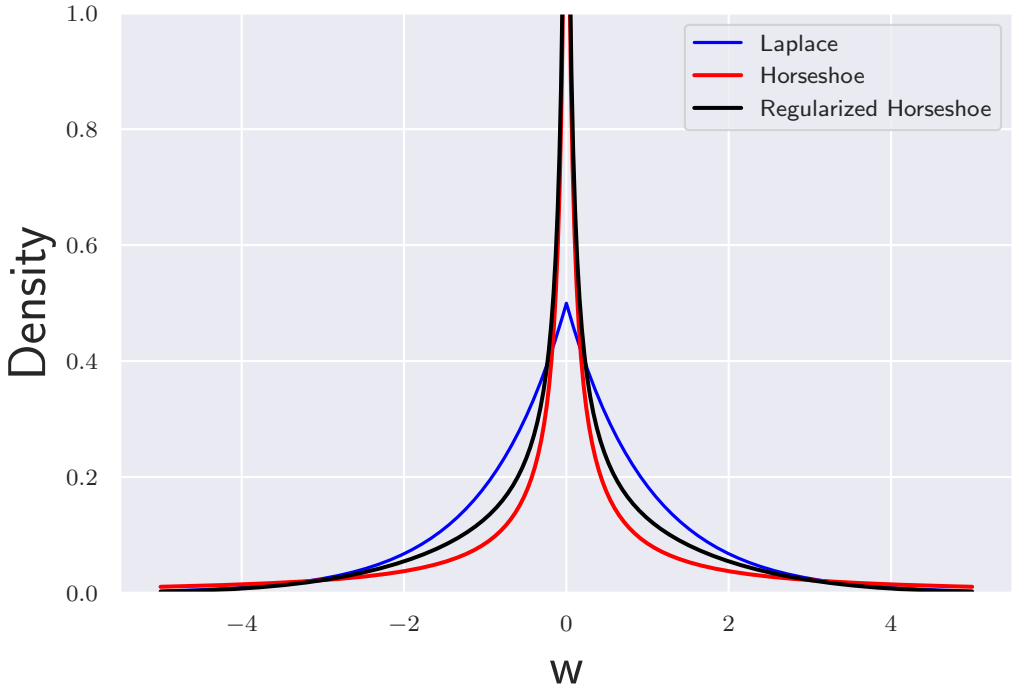


Figure 24: Density for the Laplace distribution ($\frac{1}{2b}e^{-\frac{|x|}{b}}$, with $b = 1$), the horseshoe ($b_0 = b_g = 1$), and the regularized horseshoe distribution ($b_0 = b_g = 1$ and $c = 2$). Both horseshoe variants have significantly larger spikes at zero than the Laplace distribution and the regularized horseshoe has truncated tails.

### E.1. Laplace

The Laplace distribution results in the widely used $\ell_1$ penalty and leads to sparse MAP solutions. It however exhibits thinner tails and drastically smaller spike at zero than its horseshoe counterparts. See Bhadra et al. (2017) for a more in-depth discussion contrasting horseshoe like distributions against the Laplace.

$$w \mid b \sim \text{Laplace}(w \mid 0, b)$$
$$\text{Laplace}(w \mid 0, b) = \frac{1}{2b}e^{-\frac{|w|}{b}}$$

### E.2. Horseshoe

$$w_{kl} \mid \tau_{kl}, \upsilon_l \sim \mathcal{N}(0, (\tau_{kl}^2 \upsilon_l^2)\mathbb{I}),$$
$$\tau_{kl} \sim C^+(0, b_0), \quad \upsilon_l \sim C^+(0, b_g).$$

### E.3. Regularized Horseshoe

$$w_{kl} \mid \tau_{kl}, \upsilon_l, c \sim \mathcal{N}(0, (\tilde{\tau}_{kl}^2 \upsilon_l^2)\mathbb{I}), \tilde{\tau}_{kl}^2 = \frac{c^2 \tau_{kl}^2}{c^2 + \tau_{kl}^2 \upsilon_l^2},$$
$$\tau_{kl} \sim C^+(0, b_0), \quad \upsilon_l \sim C^+(0, b_g).$$

## References

Ryan P. Adams, Hanna M Wallach, and Zoubin Ghahramani. Learning the structure of deep sparse graphical models. In *Conference on Artificial Intelligence and Statistics (AISTATS)*, 2010.

Jose M Alvarez and Mathieu Salzmann. Learning the number of neurons in deep networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

Michael Betancourt. A general metric for Riemannian manifold Hamiltonian Monte Carlo. In *International Conference on Geometric Science of Information*, pages 327–334. Springer, 2013.

Michael Betancourt and Mark Girolami. Hamiltonian Monte Carlo for hierarchical models. *Current trends in Bayesian methodology with applications*, 79:30, 2015.

Anindya Bhadra, Jyotishka Datta, Nicholas G Polson, and Brandon T Willard. Lasso meets horseshoe: A survey. *arXiv preprint arXiv:1706.10179*, 2017.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. In *International Conference on Machine Learning (ICML)*, 2015.

Wray L Buntine and Andreas S Weigend. Bayesian back-propagation. *Complex systems*, 5 (6):603–643, 1991.

Carlos M. Carvalho, Nicholas G. Polson, and James G. Scott. Handling sparsity via the horseshoe. In *Conference on Artificial Intelligence and Statistics (AISTATS)*, 2009.

Tianqi Chen, Emily Fox, and Carlos Guestrin. Stochastic gradient hamiltonian monte carlo. In *International conference on machine learning (ICML)*, 2014.

Stefan Depeweg, José Miguel Hernández-Lobato, Finale Doshi-Velez, and Steffen Udluft. Learning and policy search in stochastic dynamical systems with Bayesian neural networks. *International Conference on Learning Representations (ICLR)*, 2017.

Simon Duane, Anthony D Kennedy, Brian J Pendleton, and Duncan Roweth. Hybrid Monte Carlo. *Physics letters B*, 195(2):216–222, 1987.

Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning (ICML)*, 2016.

Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep Bayesian active learning with image data. In *Workshop on Bayesian Deep Learning, NIPS*, 2016a.

Yarin Gal, Rowan McAllister, and Carl Edward Rasmussen. Improving PILCO with Bayesian neural network dynamics models. In *Workshop on Data-Efficient Machine Learning, ICML*, 2016b.

Soumya Ghosh and Finale Doshi-Velez. Model selection in Bayesian neural networks via horseshoe priors. *Workshop on Bayesian Deep Learning, NIPS*, 2017a.

Soumya Ghosh and Finale Doshi-Velez. Model selection in Bayesian neural networks via horseshoe priors. *https://arxiv.org/abs/1705.10388*, 2017b.

Soumya Ghosh, Jiayu Yao, and Finale Doshi-Velez. Structured variational learning of Bayesian neural networks with horseshoe priors. *International Conference on Machine Learning (ICML)*, 2018.

Mark Girolami and Ben Calderhead. Riemann manifold Langevin and Hamiltonian Monte Carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2):123–214, 2011.

Tilmann Gneiting, Fadoua Balabdaoui, and Adrian E Raftery. Probabilistic forecasts, calibration and sharpness. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(2):243–268, 2007.

Arjun K Gupta and Daya K Nagar. *Matrix variate distributions*, volume 104. CRC Press, 1999.

Danijar Hafner, Dustin Tran, Alex Irpan, Timothy Lillicrap, and James Davidson. Reliable uncertainty estimates in deep neural networks using noise contrastive priors. In *Uncertainty in Artificial Intelligence (UAI)*, 2019.

Thomas M Hamill. Interpretation of rank histograms for verifying ensemble forecasts. *Monthly Weather Review*, 129(3):550–560, 2001.

Babak Hassibi, David G Stork, and Gregory J Wolff. Optimal brain surgeon and general network pruning. In *Neural Networks, 1993., IEEE Intl. Conf. on*. IEEE, 1993.

Jose Hernandez-Lobato, Yingzhen Li, Mark Rowland, Thang Bui, Daniel Hernández-Lobato, and Richard Turner. Black-box alpha divergence minimization. In *International Conference on Machine Learning (ICML)*, 2016.

José Miguel Hernández-Lobato and Ryan P. Adams. Probabilistic backpropagation for scalable learning of Bayesian neural networks. In *International Conference on Machine Learning (ICML)*, 2015.

John B Ingraham and Debora S Marks. Variational inference for sparse and undirected models. In *International Conference on Machine Learning (ICML)*, 2017.

Pavel Izmailov, Wesley J Maddox, Polina Kirichenko, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Subspace inference for Bayesian deep learning. In *Uncertainty in Artificial Intelligence (UAI)*, 2019.

Norman Lloyd Johnson, Samuel Kotz, and Narayanaswamy Balakrishnan. *Continuous univariate distributions*, volume 1. Houghton Mifflin Boston, 1970.

Ajjen Joshi, Soumya Ghosh, Margrit Betke, Stan Sclaroff, and Hanspeter Pfister. Personalizing gesture recognition using hierarchical Bayesian neural networks. In *Computer Vision and Pattern Recognition (CVPR)*, 2017.

Taylor W Killian, Samuel Daulton, Finale Doshi-Velez, and George Konidaris. Robust and efficient transfer learning with hidden parameter markov decision processes. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

Diederik P Kingma and Max Welling. Stochastic gradient VB and the variational auto-encoder. In *International Conference on Learning Representations (ICLR)*, 2014.

Diederik P. Kingma, Tim Salimans, and Max Welling. Variational dropout and the local reparameterization trick. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.

Volodymyr Kuleshov, Nathan Fenner, and Stefano Ermon. Accurate uncertainties for deep learning using calibrated regression. In *International Conference on Machine Learning (ICML)*, 2018.

Yann LeCun, John S Denker, and Sara A Solla. Optimal brain damage. In *Advances in Neural Information Processing Systems (NIPS)*, 1990.

Christos Louizos and Max Welling. Structured and efficient variational deep learning with matrix Gaussian posteriors. In *International Conference on Machine Learning (ICML)*, 2016.

Christos Louizos and Max Welling. Multiplicative normalizing flows for variational Bayesian neural networks. In *International Conference on Machine Learning (ICML)*, 2017.

Christos Louizos, Karen Ullrich, and Max Welling. Bayesian compression for deep learning. *Advances in Neural Information Processing Systems (NIPS)*, 2017.

David JC MacKay. A practical Bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472, 1992.

David JC MacKay. Developments in probabilistic modelling with neural networksensemble learning. In *Neural Networks: Artificial Intelligence and Industrial Applications*, pages 191–198. Springer, 1995.

Dougal Maclaurin, David Duvenaud, and Ryan P Adams. Autograd: Effortless gradients in numpy. In *AutoML Workshop, ICML*, 2015.

Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. Variational dropout sparsifies deep neural networks. In *International Conference on Machine Learning (ICML)*, 2017.

Iain Murray and Zoubin Ghahramani. A note on the evidence and Bayesian Occam's Razor. Technical report, Gatsby Unit Technical Report, 2005.

Kenton Murray and David Chiang. Auto-sizing neural networks: With applications to n-gram language models. *arXiv:1508.05051*, 2015.

Radford M Neal. Bayesian learning via stochastic dynamics. In *Advances in Neural Information Processing Systems (NIPS)*, 1993.

Radford M Neal. *Bayesian learning for neural networks*. PhD thesis, 1997.

Kirill Neklyudov, Dmitry Molchanov, Arsenii Ashukha, and Dmitry P Vetrov. Structured Bayesian pruning via log-normal multiplicative noise. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

Sarah E Neville, John T Ormerod, MP Wand, et al. Mean field variational bayes for continuous sparse signal shrinkage: pitfalls and remedies. *Electronic Journal of Statistics*, 8(1):1113–1151, 2014.

Tsubasa Ochiai, Shigeki Matsuda, Hideyuki Watanabe, and Shigeru Katagiri. Automatic node selection for deep neural networks using group Lasso regularization. *arXiv:1611.05527*, 2016.

Nick Pawlowski, Andrew Brock, Matthew CH Lee, Martin Rajchl, and Ben Glocker. Implicit weight uncertainty in neural networks. *arXiv preprint arXiv:1711.01297*, 2017.

Juho Piironen and Aki Vehtari. On the hyperprior choice for the global shrinkage parameter in the horseshoe prior. *Conference on Artificial Intelligence and Statistics (AISTATS)*, 2017.

Nicholas G. Polson and James G. Scott. Shrink globally, act locally: Sparse Bayesian regularization and prediction. *Bayesian Statistics*, 9:501–538, 2010.

Melanie F Pradier, Weiwei Pan, Jiayu Yao, Soumya Ghosh, and Finale Doshi-Velez. Latent projection bnns: Avoiding weight-space pathologies by learning latent representations of neural network weights. In *Workshop on Bayesian Deep Learning, NIPS*, 2018.

Rajesh Ranganath, Sean Gerrish, and David M Blei. Black box variational inference. In *Conference on Artificial Intelligence and Statistics (AISTATS)*, 2014.

Carl Edward Rasmussen and Zoubin Ghahramani. Occam's Razor. In *Advances in Neural Information Processing Systems (NIPS)*, pages 294–300, 2001.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning (ICML)*, 2014.

Hippolyt Ritter, Aleksandar Botev, and David Barber. A scalable laplace approximation for neural networks. In *International Conference on Learning Representations (ICLR)*, 2018.

Simone Scardapane, Danilo Comminiello, Amir Hussain, and Aurelio Uncini. Group sparse regularization for deep neural networks. *Neurocomputing*, 241:81–89, 2017.

Casper Kaae Sønderby, Tapani Raiko, Lars Maaløe, Søren Kaae Sønderby, and Ole Winther. Ladder variational autoencoders. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

Yale Song, David Demirdjian, and Randall Davis. Tracking body and hands for gesture recognition: NATOPS aircraft handling signals database. In *IEEE International Conference on Automatic Face & Gesture Recognition and Workshops (FG)*, 2011.

Zhao Song, Yusuke Muraoka, Ryohei Fujimaki, and Lawrence Carin. Scalable model selection for belief networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

J. T. Springenberg, A. Klein, S.Falkner, and F. Hutter. Bayesian optimization with robust bayesian neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research (JMLR)*, 15(1):1929–1958, 2014.

Shengyang Sun, Changyou Chen, and Lawrence Carin. Learning structured weight uncertainty in Bayesian neural networks. In *Conference on Artificial Intelligence and Statistics (AISTATS)*, 2017.

Shengyang Sun, Guodong Zhang, Jiaxin Shi, and Roger Grosse. Functional variational Bayesian neural networks. In *International Conference on Learning Representations (ICLR)*, 2019.

Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*, volume 1. MIT press Cambridge, 1998.

Michalis Titsias and Miguel Lázaro-gredilla. Doubly stochastic variational Bayes for non-conjugate inference. In *International Conference on Machine Learning (ICML)*, pages 1971–1979, 2014.

Matthew P Wand, John T Ormerod, Simone A Padoan, Rudolf Fuhrwirth, et al. Mean field variational Bayes for elaborate distributions. *Bayesian Analysis*, 6(4), 2011.

Ziyu Wang, Tongzheng Ren, Jun Zhu, and Bo Zhang. Function space particle optimization for Bayesian neural networks. In *International Conference on Learning Representations (ICLR)*, 2019.

Max Welling and Yee W Teh. Bayesian learning via stochastic gradient Langevin dynamics. In *International Conference on Machine Learning (ICML)*, 2011.

Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2016.

Christopher KI Williams. Computing with infinite networks. In *Advances in Neural Information Processing Systems (NIPS)*, 1997.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

Jiayu Yao, Weiwei Pan, Soumya Ghosh, and Finale Doshi-Velez. Quality of uncertainty quantification for Bayesian neural network inference. In *Workshop on Uncertainty and Robustness in Deep Learning, ICML*, 2019.