Model Selection in Bayesian Neural Networks via Horseshoe Priors

Soumya Ghosh
IBM Research AI, Cambridge, MA
ghoshso@us.ibm.com

Finale Doshi-Velez
Harvard University
finale@seas.harvard.edu

Abstract

Bayesian Neural Networks (BNNs) have recently received increasing attention for their ability to provide well-calibrated posterior uncertainties. However, model selection—even choosing the number of nodes—remains an open question. In this work, we apply a horseshoe prior over node pre-activations of a Bayesian neural network, which effectively turns off nodes that do not help explain the data. We demonstrate that our prior finds compact network structures even when the number of nodes required is grossly over-estimated. Moreover, this model selection over the number of nodes doesn't come at the expense of predictive or computational performance; in fact, we learn smaller networks with comparable predictive performance to current approaches.

1 Introduction

Bayesian Neural Networks (BNNs) are increasingly the de-facto approach for modeling stochastic functions. By treating the weights in a neural network as random variables, and performing posterior inference on these weights, BNNs can avoid overfitting in the regime of small data, provide well-calibrated 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 active learning [1, 2] and reinforcement learning [3, 4]. While there have been many recent advances in training BNNs [1, 3, 5, 6, 7], model-selection in BNNs has received relatively less attention. Unfortunately, the consequences for a poor choice of architecture are severe: too few nodes, 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 subtlely different from overfitting and under-fitting 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 2, where we see a BNN trained with too many 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 [8, 9].

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. We demonstrate that we can perform computationally-efficient and statistically-effective model selection in BNNs by placing Horseshoe priors [10] over the variance of weights incident to each node in the network. Independently of this work, Horseshoe priors were recently proposed by [11] as one of several ways to compress BNNs for fast evaluations at test time. Our work offers an additional viewpoint, in which we focus on model selection and explore opportunities for parameter-tying that dramatically decrease training time and storage requirements without sacrificing sparsity or predictive performance.

Second workshop on Bayesian Deep Learning (NIPS 2017), Long Beach, CA, USA.

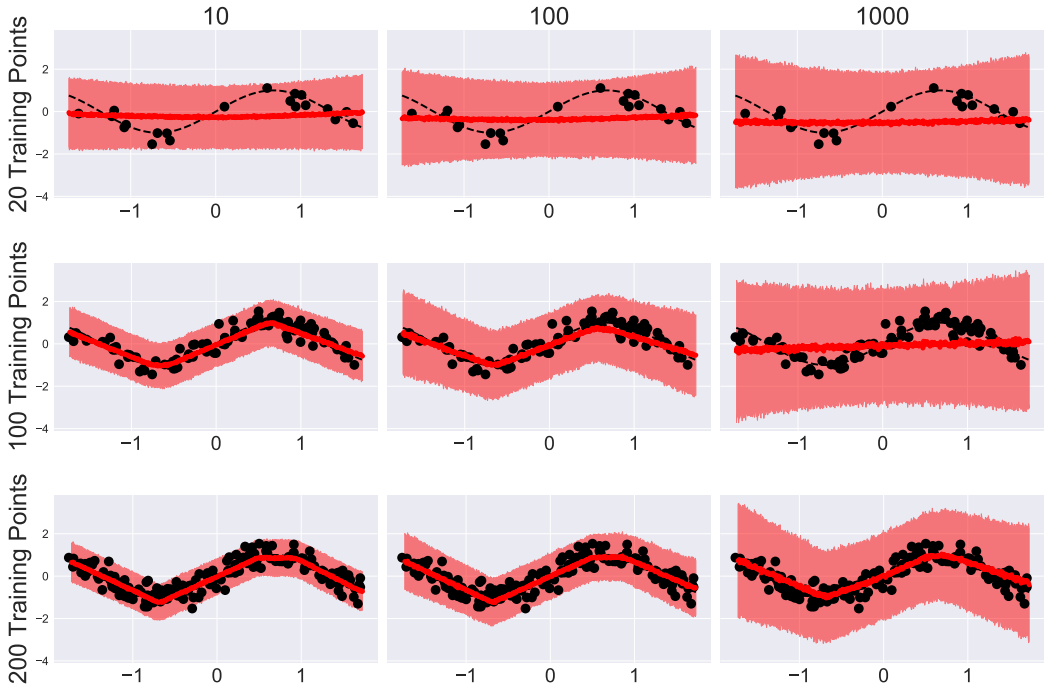


Figure 1: LEFT: Predictive distributions from a single layer BNN with varying widths and a $\mathcal{N}(0, \mathbf{I})$ prior on weights, trained on 20, 100, and 200 noisy samples from $y = sin(x) + \epsilon$, $\epsilon \sim \mathcal{N}(0, 0.1)$. With fixed data increasing BNN capacity leads to inflated uncertainty estimates and underfitting.

Automatic Model Selection through Horseshoe Priors We show that through carefully constructed priors we can induce sparsity at the unit level and prune away units that do not help explain the data well. Let $w_{kl} \in \mathbb{R}^{K_{l-1}+1}$ denote the set of all weights incident into unit k of hidden layer l. We place the following prior in w_{kl},

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

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 $a > 0$, τ_{kl} is a unit specific scale parameter, while the scale parameter υ_l is shared across the layer. This prior over weights follows the horseshoe distribution [10]. 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, Equation 1 induces sparsity at the unit level, turning off units that are unnecessary for explaining the data well.

Parameterizing for More Robust Inference: Decomposing the 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, while a Cauchy approximating family leads to high variance gradients. Instead, we use a more convenient auxiliary variable parameterization [12],

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

where $v \sim \text{Inv-Gamma}(a, b)$ is the Inverse Gamma distribution with density $p(v) \propto v^{-a-1}\text{exp}\{-b/v\}$ 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 independent 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.

Parameterizing for More Robust Inference: Non-Centered Parameterization The horseshoe prior of Equation 1 exhibits strong correlations between the weights w_{kl} and the scales $\tau_{kl}\upsilon_l$. Indeed,

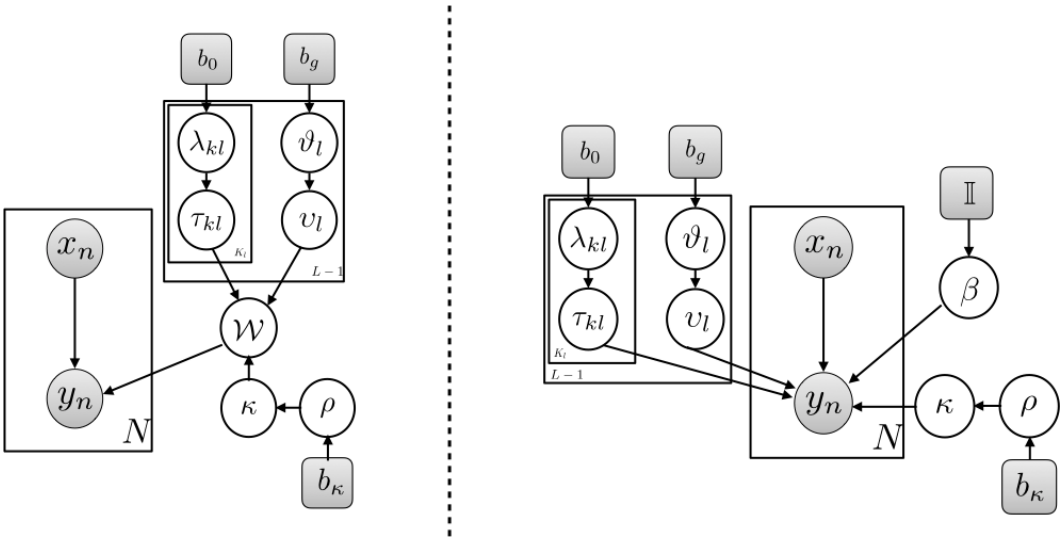


Figure 2: Graphical models capturing the conditional dependencies assumed by Bayesian Neural Networks with Horseshoe priors. Left: Centered parameterization, Right: Non-centered parameterization necessary for robust inference.

its favorable sparsity inducing properties stem from this coupling. However, an unfortunate consequence is a strongly coupled posterior that exhibits pathological funnel shaped geometries [13, 14] and is 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. Following, recent work [13, 14] we adopt a non-centered parameterization that we find empirically to significantly improve the quality of posterior approximation. We reformulate Equation 1 as $\beta_{kl} \sim \mathcal{N}(0, \mathbb{I}), \quad w_{kl} = \tau_{kl}\upsilon_l\beta_{kl}$, where the distribution on the scales are left unchanged. Figure 2 summarizes the conditional dependencies assumed by the centered and the non-centered Horseshoe BNN models.

2 Learning Bayesian Neural Networks with Horseshoe priors

We use black box variational inference to approximate the intractable posterior $p(\theta \mid \mathcal{D})$.

Approximating Family We use the following factorized variational family,

$$q(\theta \mid \phi) = q(\kappa \mid \phi_\kappa)q(\rho_\kappa \mid \phi_{\rho_\kappa}) \prod_{i,j,l} q(\beta_{ij,l} \mid \phi_{\beta_{ij,l}}); \prod_{k,l} q(\tau_{kl} \mid \phi_{\tau_{kl}})q(\lambda_{kl} \mid \phi_{\lambda_{kl}}) \prod_{l} q(\upsilon_l \mid \phi_{\upsilon_l})q(\vartheta_l \mid \phi_{\vartheta_l}). \tag{3}$$

We restrict 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 $\mathcal{N}(\beta_{ij,l} \mid \mu_{ij,l}, \sigma^2_{ij,l})$. The non-negative scale parameters τ^2_{kl} and υ^2_l and the variance of the output layer weights are constrained 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 $q(\ln \kappa^2 \mid \phi_\kappa) = \mathcal{N}(\mu_\kappa, \sigma^2_\kappa)$. The optimal variational approximations of the auxiliary variables ϑ_l, λ_{kl}, or ρ_κ follow inverse Gamma distributions. Details can be found in the supplement.

Alternate Approximating Family Choices Although, we employ a fully factorized approximation in the reparameterized space the non-centered parameterization ensures that we get a limited amount of sharing for free. To see this, observe that the variational distribution on w_{kl} implied by Equation 3 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 Ψ is a diagonal matrix with elements populated by $\sigma^2_{ij,l}$ and μ_{kl} consists of the corresponding variational means $\mu_{ij,l}$. The variational distributions of the 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 υ_l. This view immediately suggests an alternate approximating family, wherein we restrict $q(w_{kl} \mid \tau_{kl}, \upsilon_l)$ to an isotropic Gaussian by constraining

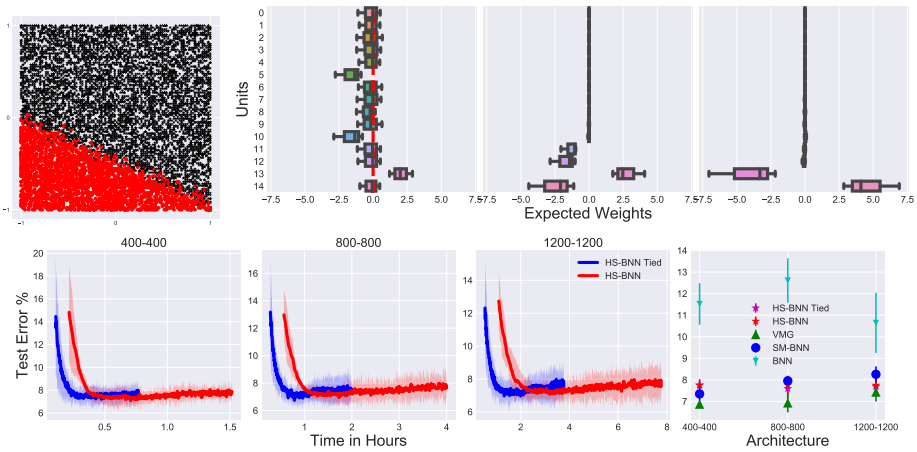


Figure 3: TOP:**Non-centered parameterization** is essential for robust inference. A synthetic nearly linear classification problem generated from sampling a 2-2-1 network. Expected weights inferred with a Bayesian neural network with Gaussian priors on weights, centered horseshoe, and a non-centered horseshoe parameterization. The boxplots display $E_q[w_{kl}]$. BOTTOM: The first three columns contrast test accuracy against training time for the tied and full approximations, averaged over 5 random splits, and run for 500 epochs each. On the right, we compare against other competing methods.

Ψ to an identity matrix. By tying the weight variances together, we effectively halve[1] the number of variational parameters to be learned and stored. Further, we demonstrate that the tied variational family leads to significant *training* speedups. We emphasize that this is complimentary to the *test-time* speedups from unit pruning.

3 Experiments

In Figure 3, we demonstrate that we demonstrate the benefits of non-centered parameterization on synthetic data, and illustrate training speed-ups afforded by parameter tying on a gesture recognition task. Additional details can be found in the supplement.

References

[1] J. M. Hernández-Lobato and R. P. Adams, "Probabilistic backpropagation for scalable learning of bayesian neural networks," in *ICML*, 2015.

[2] Y. Gal, R. Islam, and Z. Ghahramani, "Deep Bayesian active learning with image data," in *Bayesian Deep Learning workshop, NIPS*, 2016.

[3] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural networks," in *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pp. 1613–1622, 2015.

[4] S. Depeweg, J. M. Hernández-Lobato, F. Doshi-Velez, and S. Udluft, "Learning and policy search in stochastic dynamical systems with bayesian neural networks," *ICLR*, 2017.

[5] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," in *Proceedings of The 31st International Conference on Machine Learning*, pp. 1278–1286, 2014.

[6] C. Louizos and M. Welling, "Structured and efficient variational deep learning with matrix Gaussian posteriors," in *ICML*, pp. 1708–1716, 2016.

[7] J. Hernandez-Lobato, Y. Li, M. Rowland, T. Bui, D. Hernández-Lobato, and R. Turner, "Black-box alpha divergence minimization," in *ICML*, pp. 1511–1520, 2016.

[8] C. E. Rasmussen and Z. Ghahramani, "Occam's razor," in *NIPS*, pp. 294–300, 2001.

[1] Variational parameters associated with the network weights grow quadratically with network width while the scale and auxiliary variable parameters grow linearly. Storage and learning costs in large networks are dominated by the weight parameters.

[9] I. Murray and Z. Ghahramani, "A note on the evidence and bayesian occam's razor," *Gatsby Unit Technical Report*, 2005.

[10] C. M. Carvalho, N. G. Polson, and J. G. Scott, "Handling sparsity via the horseshoe," in *AISTATS*, 2009.

[11] C. Louizos, K. Ullrich, and M. Welling, "Bayesian compression for deep learning," *NIPS*, 2017.

[12] M. P. Wand, J. T. Ormerod, S. A. Padoan, R. Fuhrwirth, *et al.*, "Mean field variational Bayes for elaborate distributions," *Bayesian Analysis*, vol. 6, no. 4, pp. 847–900, 2011.

[13] M. Betancourt and M. Girolami, "Hamiltonian monte carlo for hierarchical models," *Current trends in Bayesian methodology with applications*, vol. 79, p. 30, 2015.

[14] J. B. Ingraham and D. S. Marks, "Bayesian sparsity for intractable distributions," *arXiv:1602.03807*, 2016.

[15] D. P. Kingma and M. Welling, "Stochastic gradient VB and the variational auto-encoder," in *ICLR*, 2014.

[16] R. Ranganath, S. Gerrish, and D. M. Blei, "Black box variational inference.," in *AISTATS*, pp. 814–822, 2014.

[17] M. Titsias and M. Lázaro-gredilla, "Doubly stochastic variational Bayes for non-conjugate inference," in *ICML*, pp. 1971–1979, 2014.

[18] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3-4, pp. 229–256, 1992.

[19] D. P. Kingma, T. Salimans, and M. Welling, "Variational dropout and the local reparameterization trick," in *NIPS*, 2015.

[20] D. Maclaurin, D. Duvenaud, and R. P. Adams, "Autograd: Effortless gradients in numpy," in *ICML AutoML Workshop*, 2015.

[21] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv:1412.6980*, 2014.

[22] W. L. Buntine and A. S. Weigend, "Bayesian back-propagation," *Complex systems*, vol. 5, no. 6, pp. 603–643, 1991.

[23] D. J. MacKay, "A practical Bayesian framework for backpropagation networks," *Neural computation*, vol. 4, no. 3, pp. 448–472, 1992.

[24] R. M. Neal, "Bayesian learning via stochastic dynamics," in *NIPS*, 1993.

[25] R. P. Adams, H. M. Wallach, and Z. Ghahramani, "Learning the structure of deep sparse graphical models.," in *AISTATS*, 2010.

[26] J. Piironen and A. Vehtari, "On the hyperprior choice for the global shrinkage parameter in the horseshoe prior," *AISTATS*, 2017.

[27] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian approximation: Representing model uncertainty in deep learning," in *ICML*, 2016.

[28] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting.," *JMLR*, vol. 15, no. 1, pp. 1929–1958, 2014.

[29] D. Molchanov, A. Ashukha, and D. Vetrov, "Variational dropout sparsifies deep neural networks," *arXiv:1701.05369*, 2017.

[30] Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *NIPS*, pp. 598–605, 1990.

[31] B. Hassibi, D. G. Stork, and G. J. Wolff, "Optimal brain surgeon and general network pruning," in *Neural Networks, 1993., IEEE Intl. Conf. on*, pp. 293–299, IEEE, 1993.

[32] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," in *NIPS*, pp. 2074–2082, 2016.

[33] T. Ochiai, S. Matsuda, H. Watanabe, and S. Katagiri, "Automatic node selection for deep neural networks using group lasso regularization," *arXiv:1611.05527*, 2016.

[34] S. Scardapane, D. Comminiello, A. Hussain, and A. Uncini, "Group sparse regularization for deep neural networks," *Neurocomputing*, vol. 241, pp. 81–89, 2017.

[35] K. Murray and D. Chiang, "Auto-sizing neural networks: With applications to n-gram language models," *arXiv:1508.05051*, 2015.

[36] C. K. Sønderby, T. Raiko, L. Maaløe, S. K. Sønderby, and O. Winther, "Ladder variational autoencoders," in *NIPS*, pp. 3738–3746, 2016.

[37] Y. Song, D. Demirdjian, and R. Davis, "Tracking body and hands for gesture recognition: Natops aircraft handling signals database," in *Automatic Face & Gesture Recognition and Workshops (FG 2011), 2011 IEEE International Conference on*, pp. 500–506, IEEE, 2011.

[38] A. Joshi, S. Ghosh, M. Betke, S. Sclaroff, and H. Pfister, "Personalizing gesture recognition using hierarchical bayesian neural networks," in *CVPR*, 2017.

A Black Box Variational Inference

Recent progress in black box variational inference [15, 5, 16, 17] provides a recipe for subverting this difficulty. These techniques provide 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 [18, 16] or reparameterization gradients [15, 5, 17] to differentiate through the sampling process. With the unbiased gradients 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,

$$\begin{aligned} &\nabla_{\mu,\sigma} \mathbb{E}_{q_w}[g(w)] \Leftrightarrow \nabla_{\mu,\sigma} \mathbb{E}_{\mathcal{N}(\epsilon|0,1)}[g(\mu + \sigma\epsilon)] \\ &\approx \frac{1}{S} \sum_s \nabla_{\mu,\sigma} g(\mu + \sigma\epsilon^{(s)}), \end{aligned} \tag{4}$$

for any differentiable function g and $\epsilon^{(s)} \sim \mathcal{N}(0,1)$. Further, as shown in [19], the variance of the gradient estimator can be provably lowered by noting that the weights in a layer only affect $\mathcal{L}(\phi)$ through the layer's pre-activations and directly sampling from the relatively lower-dimensional variational posterior over pre-activations.

Variational distribution on pre-activations Recall that the pre-activation of node k in layer l, u_{kl} in our non-centered model is $u_{kl} = \tau_{kl} \upsilon_l \beta_{kl}^T [a, 1]^T$. The variational posterior for the pre-activations is given by,

$$\begin{aligned} q(u_{kl} \mid \mu_{u_{kl}}, \sigma^2_{u_{kl}}) &= \mathcal{N}(u_{kl} \mid \mu_{u_{kl}}, \sigma^2_{u_{kl}}), \\ \mu_{u_{kl}} = \tau_{kl}^{(s)} \upsilon_l^{(s)} \mu_{\beta_{kl}}^T a; \quad \sigma^2_{u_{kl}} &= {\tau_{kl}^{(s)}}^2 {\upsilon_l^{(s)}}^2 \sigma^{2^T}_{\beta_{kl}} a^2, \end{aligned} \tag{5}$$

where a is the input to layer l, $\mu_{\beta_{kl}}$ and $\sigma^2_{\beta_{kl}}$ are the means and variances of the variational posterior over weights incident into node k, and a^2 denotes a point wise squaring of the input a, and $\tau_{kl}^{(s)}$, $\upsilon_l^{(s)}$ are samples from the corresponding posteriors. Since, the variational posteriors of τ_{kl} and υ_l are restricted to the log-Normal family, their product $\tau_{kl}\upsilon_l$ is another log-Normal random variable and can be directly sampled.

Algorithm We now have all the tools necessary for optimizing Equation **??**. By recursively sampling from the variational posterior of Equation 5 for each layer of the network, we are able to forward propagate information through the network. Owing to the reparameterizations (Equation 4), we are also able to differentiate through the sampling process and use reverse mode automatic differentiation tools [20] to compute the relevant gradients. With the gradients in hand, we optimize $\mathcal{L}(\phi)$ with respect to the variational weights ϕ_β, per-unit scales $\phi_{\tau_{kl}}$, per-layer scales ϕ_{υ_l}, and the variational scale for the output layer weights, ϕ_κ using Adam [21]. Conditioned on these, the optimal variational posteriors of the auxiliary variables ϑ_l, λ_{kl}, and ρ_κ 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,

$$q(\lambda_{kl} \mid \phi_{\lambda_{kl}}) = \text{Inv-Gamma}(\lambda_{kl} \mid 1, \mathbb{E}[\frac{1}{\tau_{kl}}] + \frac{1}{b_0^2}). \tag{6}$$

The distributions of the other auxiliary variables are analogous. See supplement for derivations. The overall algorithm (see Algorithm 1, involves cycling between gradient and fixed point updates to maximize the ELBO in a coordinate ascent fashion. Observe that all parameters are learned jointly.

Computational Efficiency Just as with standard back propagation, each iteration of the proposed black-box variational inference algorithm involves a forward pass and a backward pass. The computational complexity of the forward pass is identical to that of backprop and standard VI-based BNN learning. The backward-pass for a vanilla BNN is about two times more expensive than for a NN, stemming from having to update both weight means and weight variances.

Algorithm 1 HS-BNN Training

Input Model $p(\mathcal{D}, \theta)$, variational approximation $q(\theta \mid \phi)$, number of iterations T.
Output: Variational parameters ϕ
Initialize variational parameters ϕ.
for `T iterations` **do**
    Update ϕ_β, $\phi_{\tau_{kl}}$, ϕ_{υ_l}, $\phi_\kappa \leftarrow \text{ADAM}(\mathcal{L}(\phi))$.
    Conditioned on $\phi_{\tau_{kl}}$, ϕ_{υ_l}, and ϕ_κ update ϕ_{ϑ_l}, $\phi_{\lambda_{kl}}$, ϕ_{ρ_κ} using Equation 6.

The HS-BNN employing the fully factorized variational approximation has only a few additional parameters compared to the vanilla BNN: there exist scale and auxiliary random variables per unit, and each layer also has a global scale and auxiliary random variable. The computational cost of updating these is dwarfed by the cost of weight updates, and so our training is only marginally more expensive than vanilla BNNs. The tied variational approximation has far *fewer* parameters than vanilla BNNs for moderate to large networks and is consequently faster to train.

B Fixed point updates

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

$$\begin{aligned}
\mathcal{L}(\phi) &= \mathbb{E}[\ln \text{Inv-Gamma}(\kappa \mid 1/2, 1/\rho_\kappa)] + \mathbb{E}[\ln \text{Inv-Gamma}(\rho_\kappa \mid 1/2, 1/b_\kappa^2)] \\
&+ \sum_n \mathbb{E}[\ln p(y_n \mid \beta, \mathcal{T}, \kappa, x_n)] \\
&+ \sum_{l=1}^{L-1} \sum_{k=1}^{K_L} \mathbb{E}[\ln \text{Inv-Gamma}(\tau_{kl} \mid 1/2, 1/\lambda_{kl})] \mathbb{E}[\ln \text{Inv-Gamma}(\lambda_{kl} \mid 1/2, 1/b_0^2)] \\
&+ \sum_{l=1}^{L-1} \mathbb{E}[\ln \text{Inv-Gamma}(\upsilon_l \mid 1/2, 1/\vartheta_l)] + \mathbb{E}[\ln \text{Inv-Gamma}(\vartheta_l \mid 1/2, 1/b_g^2)] \\
&+ \sum_{l=1}^{L-1} \sum_{k=1}^{K_l} \mathbb{E}[\ln \mathcal{N}(\beta_{kl} \mid 0, \mathbb{I})] + \sum_{k=1}^{K_L} \mathbb{E}[\ln \mathcal{N}(\beta_{kL} \mid 0, \mathbb{I})] + \mathbb{H}[q(\theta \mid \phi)].
\end{aligned} \tag{7}$$

With our choices of the variational approximating families, all the entropies are available in closed form. We rely on a Monte-Carlo estimates to evaluate the expectation involving the likelihood $\mathbb{E}[\ln p(y_n \mid \beta, \mathcal{T}, \kappa, x_n)]$.

The auxiliary variables ρ_κ, ϑ_l and ϑ_l all follow inverse Gamma distributions. Here we derive for λ_{kl}, the others follow analogously. Consider,

$$\begin{aligned}
\ln q(\lambda_{kl}) &\propto \mathbb{E}_{-q_{\lambda_{kl}}}[\ln \text{Inv-Gamma}(\tau_{kl} \mid 1/2, 1/\lambda_{kl})] + \mathbb{E}_{-q_{\lambda_{kl}}}[\ln \text{Inv-Gamma}(\lambda_{kl} \mid 1/2, 1/b_0^2)], \\
&\propto (-1/2 - 1/2 - 1) \ln \lambda_{kl} - (\mathbb{E}[1/\tau_{kl}] + 1/b_0^2)(1/\lambda_{kl}),
\end{aligned} \tag{8}$$

from which we see that,

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

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

C Related Work

Early work on Bayesian neural networks can be traced back to [22, 23, 24]. These early approaches relied on Laplace approximation or Markov Chain Monte Carlo (MCMC) for inference. They do not

scale well to modern architectures or the large datasets required to learn them. Recent advances in stochastic variational methods [3, 5], black-box variational and alpha-divergence minimization [7, 16], and probabilistic backpropagation [1] have reinvigorated interest in BNNs by allowing inference to scale to larger architectures and data.

Work on learning structure in BNNs remains relatively nascent. In [25] the authors use a cascaded Indian buffet process to learn the structure of sigmoidal belief networks. While interesting, their approach appears susceptible to poor local optima and their proposed Markov Chain Monte Carlo based inference does not scale well. More recently, [3] 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. Their goal of turning off edges is very different than our approach, which performs model selection over the appropriate number of nodes. Our proposed Horseshoe prior instead employs an infinite scale mixture-of-Gaussians. Beyond providing stronger sparsity, this is attractive because it obviates the need to directly specify mixture component variances or the mixing proportion as required by the prior proposed in [3]. Only the prior scales of the variances needs to be specified and in our experiments, we found results to be relatively robust to the values of these hyper-parameters. Recent work [26] indicates that further gains may be possible by a more careful tuning of the scale parameters. Others [19, 27] have noticed connections between Dropout [28] and approximate variational inference. In particular, [29] 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. This is an interesting but orthogonal approach, wherein sparsity stems from variational optimization instead of the prior.

There also exists work on learning structure in non-Bayesian neural networks. Early work [30, 31] pruned networks by analyzing second-order derivatives of the objectives. More recently, [32] describe applications of structured sparsity not only for optimizing filters and layers but also computation time. Closer to our work in spirit, [33], [34] and [35] who use group sparsity to prune groups of weights—e.g. weights incident to a node. However, these approaches don't model the uncertainty in weights and provide uniform shrinkage to all parameters. Our horseshoe prior approach similarly provides group shrinkage while still allowing large weights for groups that are active. As mentioned, [11] recently used horseshoe priors for model compression.

D Experiments

In this section, we present experiments that evaluate various aspects of the proposed Bayesian neural network with horseshoe priors (HS-BNN). We begin with experiments on synthetic data that showcase the model's ability to guard against under fitting and recover the underlying model. We then proceed to a challenging gesture recognition task, and standard classification, and regression benchmarks. For regression problems we use Gaussian likelihoods with an unknown precision γ, $p(y_n \mid f(\mathcal{W}, x_n), \gamma) = \mathcal{N}(y_n \mid f(\mathcal{W}, x_n), \gamma^{-1})$. We place a vague prior on the precision,$\gamma \sim \text{Gamma}(6, 6)$ and approximate the posterior over γ using a Gamma distribution. The corresponding variational parameters are learned via a gradient update during learning. We use a Categorical likelihood for the classification problems. In a preliminary study, we found larger mini-batch sizes improved classification performance, and in all experiments we use a batch size of 512. For the regression experiments we used a batch size of 128. The hyper parameters b_0 and b_g are both set to one.

Regression Experiments We follow the experimental protocol proposed in [1, 6] and train a single hidden layer network with 50 rectified linear units for all but the larger "Protein" and "Year" datasets for which we train a 100 unit network. For the smaller datasets 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.

Classification For the classification and gesture recognition experiments we used Adam with a learning rate of 0.005 and 500 epochs. We used the parameter settings recommended in the original papers for the competing methods.

MNIST We preprocessed the images in the MNIST digits dataset by dividing the pixel values by 126. We explored networks with varying widths and depths all employing rectified linear units. We did not use a validation set to monitor validation performance or tune hyper-parameters. For

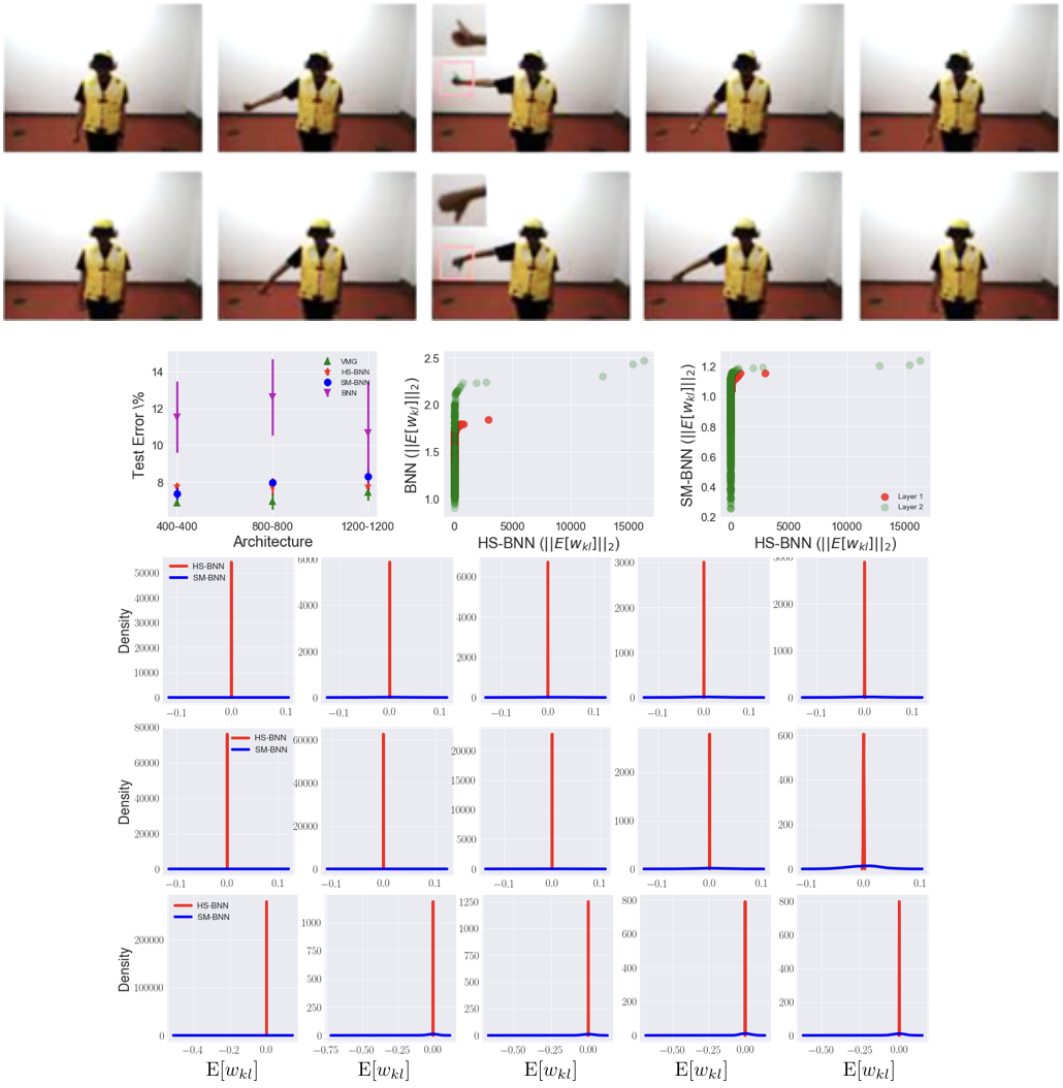
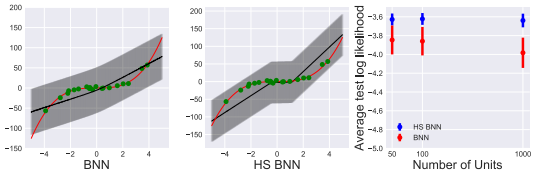


Figure 4: Gesture recognition. Top: Example gestures from the NATOPS dataset. Second Row: *Left*: Test error rates averaged over 5 runs achieved by competing methods. *Right*: Scatter plots of solutions recovered by HS-BNN and competing methods for the 800-800 architecture on one of the five splits. The bottom three rows provide density plots for the smallest 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. The plots are sorted by 2-norm of w_{kl}, from left to right.

the larger classification experiments, we found that annealing in the entropy [36] 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.

NATOPS We also experimented with a gesture recognition dataset [37] 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 4. The dataset 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. [37] are provided as features for all frames of all



Dataset	N(d)	VMG(RMSE)	HS-BNN (RMSE)	VMG(Test ll)	HS-BNN(Test ll)
Boston	506 (13)	$\mathbf{2.72 \pm 0.13}$	3.32 ± 0.66	$\mathbf{-2.48 \pm 0.66}$	-2.54 ± 0.15
Concrete	1030 (8)	$\mathbf{4.88 \pm 0.12}$	5.66 ± 0.41	$\mathbf{-3.01 \pm 0.66}$	-3.09 ± 0.06
Energy	768 (8)	$\mathbf{0.54 \pm 0.02}$	1.99 ± 0.34	$\mathbf{-1.06 \pm 0.03}$	-2.66 ± 0.13
Kin8nm	8192 (8)	$\mathbf{0.08 \pm 0.00}$	$\mathbf{0.08 \pm 0.00}$	$+1.10 \pm 0.01$	$\mathbf{+1.12 \pm 0.03}$
Naval	11,934 (16)	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$	$+2.46 \pm 0.00$	$\mathbf{+5.52 \pm 0.10}$
Power Plant	9568 (4)	4.04 ± 0.04	$\mathbf{4.03 \pm 0.15}$	-2.82 ± 0.01	$\mathbf{-2.81 \pm 0.03}$
Protein	45,730 (9)	$\mathbf{4.18 \pm 0.02}$	4.39 ± 0.04	$\mathbf{-2.88 \pm 0.00}$	-2.89 ± 0.01
Wine	1599 (11)	$\mathbf{0.63 \pm 0.01}$	$\mathbf{0.63 \pm 0.04}$	$\mathbf{-0.95 \pm 0.01}$	$\mathbf{-0.95 \pm 0.05}$
Yacht	308 (6)	$\mathbf{0.70 \pm 0.05}$	1.58 ± 0.23	$\mathbf{-1.30 \pm 0.02}$	-2.33 ± 0.01
Year	515,345 (90)	$\mathbf{8.82} \pm$ NA	$9.26 \pm$ NA	$\mathbf{-3.60} \pm$ NA	$-3.63 \pm$ NA

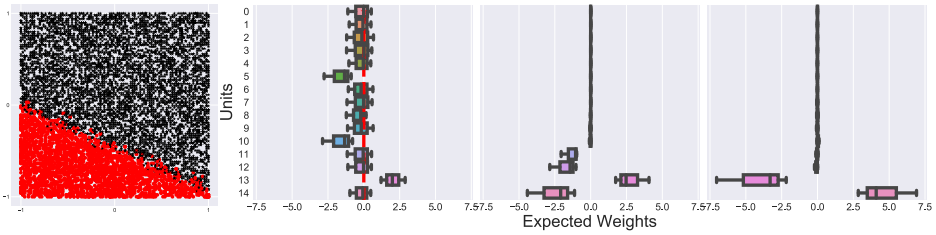


Figure 5: TOP: **Nonlinear one dimensional regression** $y_n = x_n^3 + \epsilon_n$, $\epsilon_n \sim \mathcal{N}(0, 9)$. The green dots indicate training data and the red line visualizes the noise free cubic function while the black lines are the predicted means. Left: BNN mean ± 3 standard deviations. Center: HS-BNN mean ± 3 standard deviations. Right: Average predictive log likelihood for single layer networks with 10, 100 and 1000 units. HS-BNN's performance is more robust to increasing capacity. *Right*:**Performance on UCI regression benchmarks**. BOTTOM: **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 boxplots display $E_q[w_{kl}]$.

videos in the dataset. We additionally used the 20 dimensional per-frame tracking features made available in [37]. 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.

This is a much smaller dataset than MNIST and recent work [38] has demonstrated that a BNN with Gaussian priors performs well on this task. Figure 4 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 5 random 75/25 splits of the dataset. Similar to MNIST, HS-BNN significantly outperforms BNN and is competitive with VMG and SM-BNN. We also see strong sparsity, just as in MNIST.

D.1 Experiments on simulated data

Robustness to model specification We begin with a one-dimensional non linear regression problem shown in Figure 5. To explore the effect of additional modeling capacity on performance, we sample twenty points uniformly at random in the interval $[-4, +4]$ from the function $y_n = x_n^3 + \epsilon, \epsilon \sim \mathcal{N}(0, 9)$ and train a single layer BNN with 50, 100 and 1000 units. We compare HS-BNN against a BNN with Gaussian priors on weights, $w_{ij,l} \sim \mathcal{N}(0, \kappa), \kappa \sim C^+(0, 5)$, training both for 1000 iterations. The performance of the BNN with Gaussian priors quickly deteriorates with increasing capacity conditioning on the limited amount of training data. In contrast, HS-BNN by pruning away additional capacity is more robust to model misspecification showing only a marginal drop in predictive performance with increasing number of units.

Non-centered parameterization Next, we explore the benefits of the non-centered parameterization. We consider a simple two dimensional classification problem generated by sampling data uniformly at random from $[-1, +1] \times [-1, +1]$ 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 layer on this data, with Gaussian priors, with horseshoe priors but employing a centered parameterization, and with the non-centered horseshoe prior. Each model is trained till convergence. We find that 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. In Figure 5 we visualize the posterior means of weight vectors ($\mathbb{E}[w_{kl}]$) incident onto a unit. Unsurprisingly, the BNN with Gaussian priors does

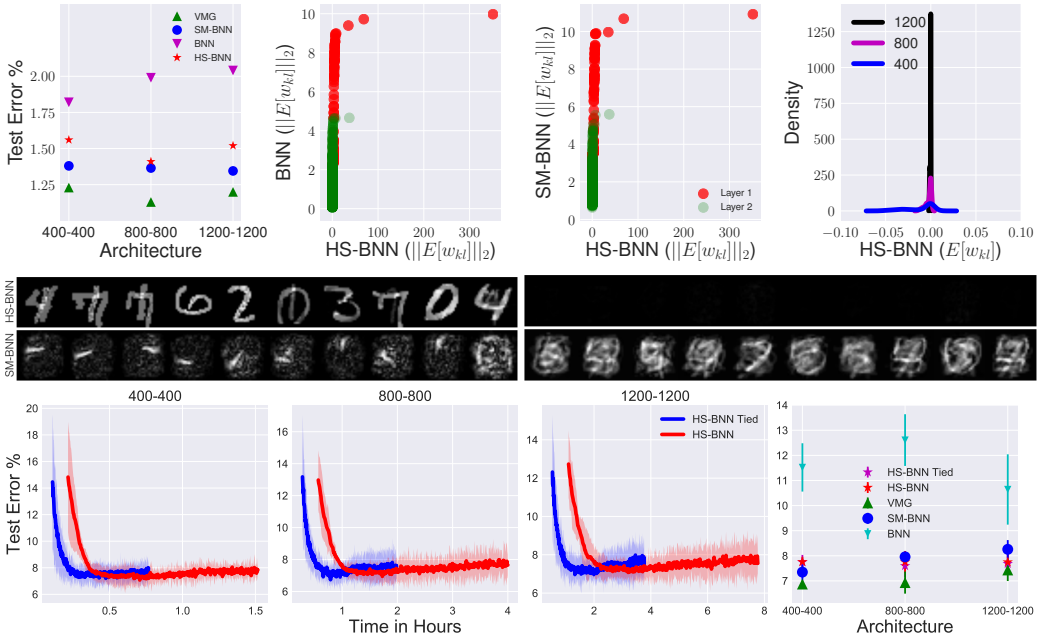


Figure 6: MNIST and NATOPS experiments. TOP: *From left to right*, Test error rates for different architectures and methods. The right two plots compare the sparsity of the solutions found by HS-BNN, SM-BNN and BNN. For the 1200-1200 network, we compare the expected node weight vectors inferred under 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 escape un-shrunk for HS-BNN. The rightmost plot shows the density of the unit with the lowest norm from the the three architectures. Middle: $\mathbb{E}[w_{kl}]$ for the first layer trained on MNIST. The left and right columns visualize the ten units with the largest and the smallest norms. BOTTOM: The first three columns contrast test accuracy against training time for the tied and full approximations, averaged over 5 random splits, and run for 500 epochs each. On the right, we compare against other competing methods.

not exhibit sparsity. 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. The non-centered parameterization however does significantly better and prunes away all but two units. Further experiments provided in the supplement demonstrate the same effect for wider 100 unit networks. The non-centered parameterized model is again able to recover the two active units.

D.2 Classification and Gesture recognition

We benchmark classification performance on the MNIST dataset and on a gesture recognition dataset (NATOPS) [37] that consists of 24 unique gestures from 20 subjects, with each subject performing 20 repetitions of each gesture. Large inter-subject variance in gestures, as well as the relatively small number of gestures, make this a challenging problem [38]. We compare HS-BNN with full and tied approximate posteriors (HS-BNN tied) against the variational matrix Gaussian (VMG) [6], a BNN with a two-component scale mixture (SM-BNN) prior on weights proposed in [3] and a BNN with a Gaussian prior (BNN) on weights. These approaches constitute the *state-of-the-art* in variational learning for BNNs.

MNIST Figure 6 summarizes our findings. We showcase results for three architectures with two hidden layers each containing 400, 800 and 1200 rectified linear hidden units. We clearly see the sparsity inducting effects of 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 sparsity, but on weights not nodes. Further, the horseshoe prior with its thicker tails and taller spike at origin encourages stronger sparsity. To see this we compared the 2-norms of the inferred expected weight node vectors $\mathbb{E}[w_{kl}]$ found by SM-BNN and HS-BNN (Figure 6). For HS-BNN the inferred scales are tiny 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 pruning the unit and not use the extra capacity. To further explore the implications of unit versus weight sparsity, we visualize $\mathbb{E}[w_{kl}]$ learned by SM-BNN and HS-BNN in Figure 6. Weight sparsity in SM-BNN encourages fundamentally different filters that pick up edges at different orientations. In contrast, HS-BNN's node sparsity encourages filters that correspond to digits or superpositions of digits and may lead to more interpretable networks. Stronger sparsity 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. Additional results exploring unit sparsity are available in the supplement.

Next, on the more challenging NATOPS dataset, we first explore the benefits afforded by the tied approximation. We performed experiments on five random replicates each using 75% of the data for training. Figure 6 contrasts the held-out test performance of the full and tied approximations against wall-clock time[2]. The tied approximation converges significantly faster nearly halving the training time without adversely affecting predictive performance. Despite the strong sparsity constraints of HS-BNN and HS-BNN tied, we find that across architectures, our performance to be significantly better than BNN, marginally better than SM-BNN, and comparable to VMG.

Regression We also benchmark performance of our model on a wide variety of regression datasets from the UCI repository. Here, we only benchmark against VMG, which has previously been shown to outperform alternatives [6]. Figure 5 summarizes our predictive results. Similar to classification and gesture recognition experiments, despite being biased towards smaller models HS-BNN remains competitive both in terms of root mean squared error (RMSE) as well as predictive log likelihoods.

E Additional Experiments

E.1 Simulated Data

Here we provide an additional experiment with the data setup in Section 6.2. We use the same linearly separable data, but train larger networks with 100 units each. Figure 7 shows the inferred weights under the different models. Observe that the non-centered HS-BNN is again able to prune away extra capacity and recover two active nodes.

[2]Experiments were performed on a 2.5 GHz Intel Core i7, with 16GB of RAM

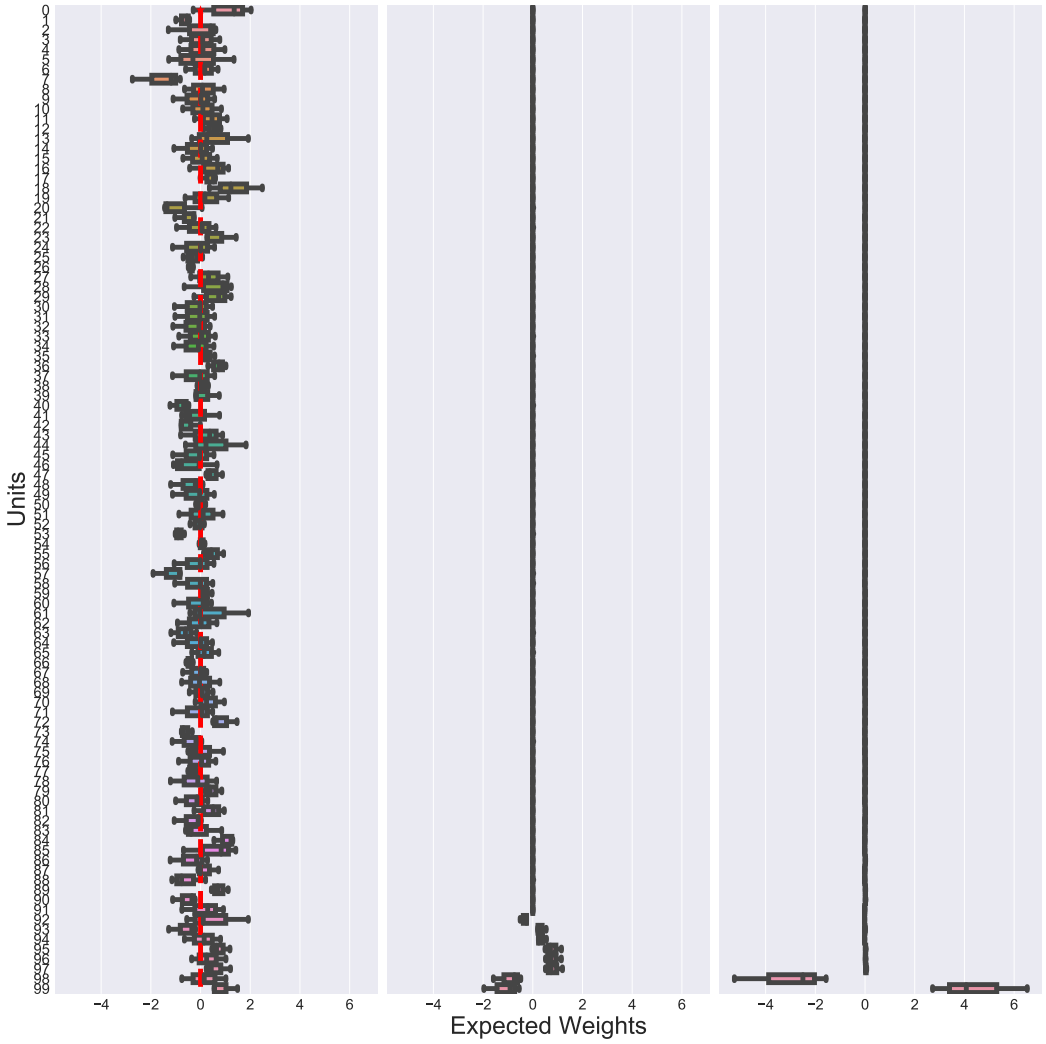


Figure 7: Learned sparsity on a synthetic classification problem. We use a single hidden layer network with 100 units. Left: Bayesian neural network with Gaussian. Center: Horseshoe, centered parameterization. Right: Horseshoe, non-centered parameterization.