Radial and Directional Posteriors for Bayesian Deep Learning*

Changyong Oh,[†1] **Kamil Adamczewski,**[1,2] **Mijung Park**[1,3]
[1]Empirical Inference Department, Max-Planck-Institute for Intelligent Systems
[2]Max Planck ETH Center for Learning System
[3]Department of Computer Science, University of Tübingen
{changyong.om, mijung.park}@tuebingen.mpg.de, kamil.m.adamczewski@gmail.com

Abstract

We propose a new variational family for Bayesian neural networks. We decompose the variational posterior into two components, where the *radial* component captures the strength of each neuron in terms of its magnitude; while the *directional* component captures the statistical dependencies among the weight parameters. The dependencies learned via the directional density provide better modeling performance compared to the widely-used Gaussian mean-field-type variational family. In addition, the strength of input and output neurons learned via our posterior provides a structured way to compress neural networks. Indeed, experiments show that our variational family improves predictive performance and yields compressed networks simultaneously.

1 Introduction

Neural networks have recently become revolutionary tools to solve numerous statistical problems in science and industry. However, the uncertainty of the network weight estimates is often neglected, although there is a growing necessity to model the uncertainty in many application domains such as medical decision-making and climate prediction (Slingo and Palmer 2011).

Reasoning about uncertainties in the neural network models was initiated by a few seminal papers (Neal 2012; MacKay 1995; Dayan and Hinton 1996). These works aimed to develop Bayesian methods for neural network models, suggesting a new research direction, called *Bayesian neural networks* (BNNs) (Blundell et al. 2015; Hernández-Lobato and Adams 2015; Gal and Ghahramani 2016; Louizos and Welling 2016). These, however, rely on prior and posterior pairs that are often chosen for convenience in inference, namely, computational tractability. The so-called *mean-field* variational family in these works assumes the posterior distributions to be all factorizing, and hence neglects the possibility of modelling statistical dependencies (i.e., correlations) among weight parameters (Graves 2011; Blundell et al. 2015; Kingma, Salimans, and Welling 2015; Neklyudov 2017; Molchanov, Ashukha, and Vetrov 2017). Capturing dependencies between the weight parameters and their uncertainties is likely to yield better models in terms of predictability. For instance, Louizos and Welling (2016) and Sun (2017) propose Gaussian posteriors with covariance matrices of certain structures such that each covariance can capture the dependencies among the input and output dimensions of each layer. Rather than proposing a new variational family, Lakshminarayanan, Pritzel, and Blundell (2017) use an ensemble of neural network models and Ritter, Botev, and Barber (2018) make use of the Laplace approximation to a trained network to obtain uncertainty estimates. In this work, we follow this train of thought and along with these papers empirically show that the methods that model weight dependencies improve prediction performance in terms of test likelihoods.

An important application of the learned uncertainty estimates is model selection via model pruning. The concept of pruning, sparsifying, or compressing a network makes sense for deep neural network models as neural network models are often over-parameterized. The task of compressing a network has received much attention due to the immense demand on the efficient deployment of deep models on mobile devices. For instance, Louizos, Ullrich, and Welling (2017) consider group Horseshoe priors and log-Normal posteriors for pruning out neurons, and Ghosh (2018) improve the result of Louizos, Ullrich, and Welling (2017) by adopting regularized Horseshoe priors and more structured posteriors. Neklyudov (2017) takes a slightly different approach which is pruning units via truncated log-Normal priors over unit scales. After removing redundant parameters, all of these methods provide significantly compressed networks.

Our contributions In this paper, we propose a new variational family with the aim of not only tackling the modelling side of BNNs in terms of capturing correlations among weight parameters but also addressing the issue of sparsification of over-parameterized neural network models. We provide a detailed description of our contributions below.

- **Proposing a novel variational family**: Our variational family has two components by decomposing a weight vector into its magnitude (radius) and angle (direction); and hence our variational prior and posterior distributions

*At https://arxiv.org/abs/1902.02603v3, supplementary material is accessible.

†Work done while at MPI-IS, now at University of Amsterdam(c.oh@uva.nl)

are decomposed into radial and directional densities. This combination of prior and posterior pairs have the following benefits. *Radial density:* We use the Half-Cauchy prior and the log-Normal posterior to provide a structured compression method for neural network parameters. The Half-Cauchy prior is chosen to provide a bias toward zero in the norm of weights. The log-Normal posterior is chosen to provide a closed-form KL divergence between prior and posterior in the computation of evidence lower bound. *Directional density:* We use the von Mises-Fisher (vMF) distribution to capture correlations between the weight parameters inspired by directional data modelling (Banerjee 2005).

- **Proposing an approximation method for numerically stable gradient estimation**: While the vMF distribution is useful for directional data modelling, applying it in variational inference poses a practical challenge due to difficulty in computing ratios of Bessel functions (not supported yet in most deep learning frameworks). We propose an approximation to the polynomials of ratios of modified Bessel functions for numerically stable gradient estimation, which is scalable to high dimensions such as several hundreds or thousands. This approximation can be applied to general variational inference problems using vMF distributions in high dimensional spheres.
- **Achieving competitive predictive and compression performance**: Equipped with these two components, our ultimate goal is to find a *good* model for a dataset, via finding the dependencies between the weights and reducing the number of parameters. So the goodness of a model will be measured by test likelihoods and prediction accuracy on test data, along with the size of the resulting model in terms of number of remaining parameters and FLOPS (Floating-Point Operations Per Second).

The rest of the paper is organized as follows. We start by describing essential quantities in variational Bayesian learning for neural network models in Sec. 2. In Sec. 3, we introduce our new variational family and address how to train the new model with high dimensional vMF in a numerically stable way. In Sec. 4, we contrast our method with other existing work. Finally in Sec. 5 we empirically show that our method improves predictive ability over existing methods together with structured pruning.

2 Variational Bayesian learning for neural networks

Consider a neural network consisting of fully-connected[1] layers with a collection of parameters $\mathbf{W} = \{\mathbf{W}^{(l)}\}_{l=1,\cdots,L}$, where the l-th layer's weight matrix is denoted by $\mathbf{W}^{(l)} \in \mathbb{R}^{n_l \times n_{l-1}}$. Here, n_l is the number of output neurons and n_{l-1} is the number of input neurons. In variational Bayesian neural networks, in an attempt to capture distributional behaviors of the weights, we often assume a tractable parametric family for the prior distribution $p_{\boldsymbol{\theta}}(\mathbf{W})$ and the approximate posterior $q_{\boldsymbol{\phi}}(\mathbf{W})$, where the parameters for each distribution are denoted by $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$, respectively. Given a dataset $\mathcal{D}$, we maximize the variational (evidence) lower bound (ELBO) to the marginal data likelihood

$$\begin{aligned}\mathcal{L}(\mathcal{D};\boldsymbol{\phi},\boldsymbol{\theta}) =& \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{W})}[\log p(\mathcal{D}|\mathbf{W})] \\ &- D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{W})||p_{\boldsymbol{\theta}}(\mathbf{W})) \qquad (1)\end{aligned}$$

in order to choose the parameters of prior and posterior distributions. The first term in Eq.1 is the expected log-likelihood with respect to the variational posterior. The second term in Eq.1 is the KL divergence between the variational posterior and the prior distribution.

We would like to point out two important conditions for the lower-bound optimization to be successful in practice. First, under neural network models, the expected log-likelihod term in Eq.1 does not have a close form and is typically estimated via Monte Carlo (MC) sampling (Hoffman 2013; Kingma and Welling 2013). The MC estimator, however, exhibits high variance in the gradients of expected log-likelihood. To alleviate this issue, many solutions have been proposed (Kingma and Welling 2013; Maddison 2016; Jang, Gu, and Poole 2016; Figurnov, Mohamed, and Mnih 2018; Tucker 2017). The most known solution among them is *reparametrization trick,* which detaches the parameters of $q_{\boldsymbol{\phi}}(\mathbf{W})$ from the randomness of $q_{\boldsymbol{\phi}}(\mathbf{W})$ so that the MC-estimate of the gradient of the expected log-likelihood can be computed stably even with the randomness involved. Using the reparametrization, we can reduce the variance of the MC-estimate of the gradient of log-likelihood. Secondly, for arbitrary priors $p(\mathbf{W})$ and posteriors $q(\mathbf{W})$, similar MC-estimates using the reparametrization trick can be found (Blundell et al. 2015). However, in practice, many models propose closed-form KL-divergence to further reduce the variance of the gradient estimates (Louizos, Ullrich, and Welling 2017). In summary, being able to use the reparameterization trick and having a closed-form KL divergence are important conditions for variational learning to be effective in BNNs. We will revisit this point in Sec. 3.

From a modelling perspective, specifying the parametric forms of the prior and posterior distributions is essential in variational BNNs. One of the most commonly used variational family is fully factorized distribution referred to as the *mean-field* variational family (Graves 2011; Blundell et al. 2015; Kingma, Salimans, and Welling 2015; Neklyudov 2017; Molchanov, Ashukha, and Vetrov 2017),

$$q(\mathbf{W}) = \prod_{l=1}^{L} \prod_{w \in \mathbf{W}^{(l)}} q(w),$$

where the posterior distribution is described by a product of distributions of each individual entry in $\mathbf{W}$. This choice is due to computational tractability. Consider the Gaussian mean-field variational family, $q(w_{i,j}^{(l)}|\boldsymbol{\phi}_{i,j}) = \mathcal{N}(\mu_{i,j}, \sigma_{i,j}^2)$, where the *variational* parameters are $\boldsymbol{\phi}_{i,j} = \{\mu_{i,j}, \sigma_{i,j}^2\}$. This formulation assumes independence between variational parameters. However, this variational family ignores any statistical correlations between the weight parameters, which

[1]For notational simplicity, we stick to a network with fully-connected layers. See Sec. 3 for details on the implementations for convolutional layers.

is the issue we address in this paper. Next we describe our new variational family which we developed taking into consideration all the aforementioned computational aspects, namely, the reparameterizability of expected log-likelihood term, a closed-form KL divergence term, and independence between variational parameters.

3 RDP: Radial and directional posterior

We propose a new variational family, which is an instance of *structured* mean-field approximation where each row (and/or column) of $\mathbf{W}^{(l)}$ is factorized,

$$q(\mathbf{W}) = \prod_{l=1}^{L} \prod_{r=1}^{n_l} q_r^{(l)}(\mathbf{w}_r^{(l)}),$$

where $\mathbf{w}_r^{(l)} \in \mathbb{R}^{n_{l-1}}$ is the r-th row of the weight matrix at l-th layer. For the sake of simplicity, we consider row-wise factorization here. But we discuss both row and column-wise factorizations at the end of this section. We decompose each factor $q_r^{(l)}$ into a density on the L_2-norm of $\mathbf{w}_r^{(l)}$ and another density on the direction of the normalized vector $\frac{\mathbf{w}_r^{(l)}}{\|\mathbf{w}_r^{(l)}\|_2}$, i.e., $q_r^{(l)}(\mathbf{w}_r^{(l)}) = q_{r,rad}^{(l)}(\|\mathbf{w}_r^{(l)}\|_2) \cdot q_{r,dir}^{(l)}\left(\frac{\mathbf{w}_r^{(l)}}{\|\mathbf{w}_r^{(l)}\|_2}\right)$, where $q_{r,rad}^{(l)}$ is the *radial density* and $q_{r,dir}^{(l)}$ is the *directional density*.

Directional density We take the prior $p_{r,dir}^{(l)}$ and the posterior $q_{r,dir}^{(l)}$ distributions to be the *von Mises-Fisher* (vMF) distribution (Mardia and Jupp 2009), which is a probability density on a unit (hyper)sphere

$$q_{r,dir}^{(l)}\left(\frac{\mathbf{w}_r^{(l)}}{\|\mathbf{w}_r^{(l)}\|_2}\right) = vMF\left(\frac{\mathbf{w}_r^{(l)}}{\|\mathbf{w}_r^{(l)}\|_2}; \boldsymbol{\mu}_r^{(l)}, \kappa_r^{(l)}\right), \quad (2)$$

where $vMF(\mathbf{x}; \boldsymbol{\mu}, \kappa) = C_d(\kappa) \exp(\kappa \boldsymbol{\mu}^T \mathbf{x})$, where $C_d(\kappa) = \frac{\kappa^{d/2-1}(\kappa)}{(2\pi)^{d/2} \mathcal{I}_{d/2-1}(\kappa)}$. The location parameter $\boldsymbol{\mu}$ is also a d-dimensional unit vector, κ is the concentration parameter, and $I_{d/2-1}$ is the modified Bessel function of the first kind at order $d/2-1$. The vMF distribution is intuitively understood as multivariate Gaussian distribution with a diagonal covariance matrix on unit (hyper)sphere.

In our prior and posterior distributions, we assume that the concentration parameter is shared across all the rows in each layer, by assigning a single concentration parameter, $\kappa_r^{(l)} = \kappa^{(l)}$ for all $r = \{1, \cdots, n_l\}$, while the mean vector parameters are separately assigned for each row. This way we can reduce the number of variational and prior parameters significantly. The resulting prior $p_{r,dir}^{(l)}$ and posterior $q_{r,dir}^{(l)}$ distributions have the following forms:

$$p_{r,dir}^{(l)}\left(\frac{\mathbf{w}_r^{(l)}}{\|\mathbf{w}_r^{(l)}\|_2}\right) = vMF\left(\boldsymbol{\mu}_{p,r}^{(l)}, \kappa_p^{(l)}\right),$$
$$q_{r,dir}^{(l)}\left(\frac{\mathbf{w}_r^{(l)}}{\|\mathbf{w}_r^{(l)}\|_2}\right) = vMF\left(\boldsymbol{\mu}_{q,r}^{(l)}, \kappa_q^{(l)}\right), \quad (3)$$

where the prior mean parameters and the concentration parameter are denoted by $\boldsymbol{\mu}_{p,r}^{(l)}$ and $\kappa_p^{(l)}$, respectively, and those in the posterior distribution are denoted by $\boldsymbol{\mu}_{q,r}^{(l)}$, and $\kappa_q^{(l)}$. Explicitly modelling the directional component using vMF allows us to capture the dependence within the weight parameters of each row. Having the same concentration parameter across all the rows within each layer induces a particular way of dependence in the weight parameters within the same layer. If the mean parameters of each row's weights are close to each other, having the same concentration level, possibly a high concentration level (which we expect if the posterior confidence is high) makes the row-wise directional densities *more similar* to each other, and vice versa. This particular way of parameterizing the variational parameters allows us to capture the dependence across rows without assigning concentration parameters to each of the rows and layers separately.

Radial density While we could adopt any probability distribution with a non-negative support for the radial density, we focus on distributions that can promote sparsity in the resulting posterior. Specifically, inspired by the group horseshoe prior proposed in (Louizos, Ullrich, and Welling 2017), we take a product of two Half-Cauchy distributions to be our prior in order to induce sparsity in the *norms* of the weights. First, we write down the norm of each row given a layer as a product of two independent *half-Cauchy* random variables, $\|\mathbf{w}_r^{(l)}\|_2 = s^{(l)} z_r^{(l)}$, where $s^{(l)} \sim \mathcal{C}^+(\gamma)$, $z_r^{(l)} \sim \mathcal{C}^+(1)$, and the prior is given by

$$p_{r,rad}^{(l)}(\|\mathbf{w}_r^{(l)}\|_2) = \mathcal{C}^+(s^{(l)}; \gamma) \cdot \mathcal{C}^+(z_r^{(l)}; 1), \quad (4)$$

where the probability density function for a half-Cauchy distributed random variable x is given by $\mathcal{C}^+(x; \gamma) = \frac{2}{\pi\gamma(1+(x/\gamma)^2)}$, with a scale parameter $\gamma > 0$. The smaller the scale parameter gets, the larger the probability mass concentrates around zero. At this point, it might not be immediately clear why we chose to use two Half-Cauchy distributions as a prior rather than one. Our explanation is as follows.

What we ultimately hope to control is the level of sparsity in the weights drawn from the resulting posterior distribution. We allow the posterior to have two different levels of sparsity, namely, *local* (row-wise) sparsity and *global* (layer-wise) sparsity. The reason we write the norm as a product of two terms $\|\mathbf{w}_r^{(l)}\|_2 = s^{(l)} z_r^{(l)}$ is that each of these terms affects the local sparsity via $z_r^{(l)}$ and global sparsity via $s^{(l)}$ in the posterior distribution, respectively. Even when all radii are small, the largest one among them has significant influence in model performance. Thus, we can use the relative strength of radius densities to prune out.

Before describing our radial posterior density, we need to elaborate on the fact that each of half-Cauchy distributions can be further factorized. As used in (Louizos, Ullrich, and Welling 2017), the half-Cauchy distribution can be written as a product of an inverse-Gamma density and a Gamma density due to the fact that the square of half-Cauchy $\mathcal{C}^+$ is equal in distribution to a product of Gamma and inverse-Gamma (Neville 2014). Hence, we rewrite the two factors

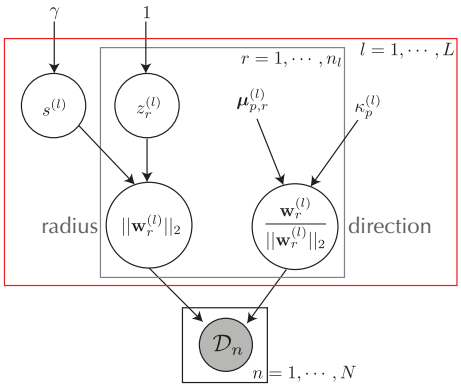

Figure 1: Graphical representation of our generative model. The black rectangle describes row parameters and red rectangle layer parameters. The radius part is controlled by two half-Cauchy random variables $s^{(l)}$ and $z_r^{(l)}$, while the directional part is controlled by a vMF distribution governed by the prior mean vector and a concentration parameter, $\boldsymbol{\mu}_r^{(l)}$ and $\kappa_p^{(l)}$, respectively. The two parts taken together govern the data probability (bottom).

$z_r^{(l)}$ and $s^{(l)}$ for describing $\|\mathbf{w}_r^{(l)}\|_2$ as

$$\sqrt{z_r^{(l)}} = \alpha_r^{(l)}\beta_r^{(l)}, \text{where } \alpha_r^{(l)} \sim \mathcal{G}(\tfrac{1}{2}, 1), \beta_r^{(l)} \sim \mathcal{IG}(\tfrac{1}{2}, 1),$$
$$\sqrt{s^{(l)}} = s_a^{(l)}s_b^{(l)}, \text{where } s_a^{(l)} \sim \mathcal{G}(\tfrac{1}{2}, \gamma^2), s_b^{(l)} \sim \mathcal{IG}(\tfrac{1}{2}, 1),$$

where we denote the Gamma distribution by $\mathcal{G}$ and the inverse-Gamma distribution by $\mathcal{IG}$. In addition, Gamma (and inverse-Gamma) random variables have another nice property, that is, one can obtain a closed-form expression of KL divergence between a Gamma (and inverse-Gamma) distribution and a log-normal distribution. Using this fact, we take the log-normal distribution to be our posterior such that each of these terms can control the local sparsity and global sparsity, given as

$$q_{r,rad}^{(l)}(\|\mathbf{w}_r^{(l)}\|) = \mathcal{LN}(\alpha_r^{(l)}|\mu_r^{(l)}, \sigma_r^{2\,(l)})\mathcal{LN}(\beta_r^{(l)}|\widehat{\mu}_r^{(l)}, \widehat{\sigma}_r^{2\,(l)})$$
$$\mathcal{LN}(s_a^{(l)}|\mu^{(l)}, \sigma^{2\,(l)})\mathcal{LN}(s_b^{(l)}|\widehat{\mu}^{(l)}, \widehat{\sigma}^{2\,(l)}), \quad (5)$$

where $\mu_r^{(l)}$, $\sigma_r^{2\,(l)}$ and $\widehat{\mu}_r^{(l)}$, $\widehat{\sigma}_r^{2\,(l)}$ are the variational parameters regularized by Gamma density and inverse-Gamma density of $\mathcal{C}^+(z_r^{(l)}|1)$, respectively. Similarly, $\mu^{(l)}$, $\sigma^{2\,(l)}$ and $\widehat{\mu}^{(l)}$, $\widehat{\sigma}^{2\,(l)}$ are variational parameters for the prior $\mathcal{C}^+(s^{(l)}|\gamma)$ the same way.

Radial and directional posterior (RDP) In summary, our variational family $q(\mathbf{W})$ is given as

$$q(\mathbf{W}) = \prod_{l=1}^{L}\prod_{r=1}^{n_l} q_{r,rad}^{(l)}(\|\mathbf{w}_r^{(l)}\|) vMF\left(\frac{\mathbf{w}_r^{(l)}}{\|\mathbf{w}_r^{(l)}\|_2}; \boldsymbol{\mu}_{q,r}^{(l)}, \kappa_q^{(l)}\right) \quad (6)$$

We refer to this collection of densities as the *radial and directional posterior (RDP)*. Our prior distribution $p(\mathbf{W})$ is also factored into radius and direction as in the proposed variational family

$$p(\mathbf{W}) = \prod_{l=1}^{L}\mathcal{C}^+(s^{(l)};\gamma)\prod_{r=1}^{n_l}\mathcal{C}^+\left(z_r^{(l)};1\right)$$
$$\cdot vMF\left(\frac{\mathbf{w}_r^{(l)}}{\|\mathbf{w}_r^{(l)}\|_2}; \boldsymbol{\mu}_{p,r}^{(l)}, \kappa_p^{(l)}\right). \quad (7)$$

We denote the prior parameters collectively by $\boldsymbol{\theta} = \{\gamma, \boldsymbol{\mu}_{p,r}^{(l)}, \kappa_p^{(l)}\}$ for all layers $l = \{1, \cdots, L\}$, and the variational parameters by $\boldsymbol{\phi} = \{\mu^{(l)}, \sigma^{2\,(l)}, \widehat{\mu}^{(l)}, \widehat{\sigma}^{2\,(l)}, \mu_r^{(l)}, \sigma_r^{2\,(l)}, \widehat{\mu}_r^{(l)}, \widehat{\sigma}_r^{2\,(l)}, \boldsymbol{\mu}_{q,r}^{(l)}, \kappa_q^{(l)}\}$ for all layers $l = \{1, \cdots, L\}$ and all rows $r = \{1, \cdots, n_l\}$. A graphical representation of our model is given in Fig. 1 for the generative process.

Optimizing evidence lower bound with RDP

Recall that as far as our objective function Eq.1 is concerned, two conditions need to be met for the gradients of this objective function to well behave. The first condition (about MC estimates of the expected log-likelihood term) is whether our posterior is reparameterizable. In fact, we can represent our choice of posterior by a differentiable function $h(\epsilon, \boldsymbol{\phi})$, where the variational parameters $\boldsymbol{\phi}$ are separated from the random source, $\epsilon \sim s(\epsilon)$. See Supplement for details.

The second condition is whether the KL term is closed-form, which is the case as we choose the prior and posterior pair considering this condition. The KL term $D_{KL}(q_{\boldsymbol{\phi}}(\mathbf{W})\|p_{\boldsymbol{\theta}}(\mathbf{W}))$ is given by

$$\sum_{l}\sum_{r}^{n_l} D_{KL}(vMF(\boldsymbol{\mu}_{q,r}^{(l)}, \kappa_q^{(l)})\|vMF(\boldsymbol{\mu}_{p,r}^{(l)}, \kappa_p^{(l)}))$$
$$+D_{KL}(q_{r,rad}^{(l)}(\|\mathbf{w}_r^{(l)}\|_2)\|p_{r,rad}^{(l)}(\|\mathbf{w}_r^{(l)}\|_2)),$$

and the closed-form expressions of each of the terms are given in Supplement. Although the KL term between the vMF prior and posterior is elegantly written in closed-form,

$$D_{KL}(vMF(\mu_q, \kappa_q)\|vMF(\mu_p, \kappa_p)) =$$
$$(\kappa_q - \kappa_p\mu_p^T\mu_q)\frac{I_{d/2}(\kappa_q)}{I_{d/2-1}(\kappa_q)} + \log(C_d(\kappa_q)) - \log(C_d(\kappa_p))$$

the gradient expressions (see Supplement) with respect to the variational parameters require computing the ratio $\frac{I_{d/2}(\kappa_q)}{I_{d/2-1}(\kappa_q)}$ (Davidson 2018), which is numerically unstable. This is due to the fact that the modified Bessel function of the first kind (Bessel function) decays rapidly, so the computation of ratios of Bessel functions causes numerical errors when it tries to compute $\frac{0}{0}$ (See Supplement for detailed explanations). This gets worse with higher dimensions, and occurs even for moderate dimensions such as 50 to 100. Hence, rather than numerically computing the ratio of Bessel functions, we resort to the following Theorem.

Theorem 1 (Theorem 5 in (Ruiz-Antolín and Segura 2016))**.**

$$B_2(\nu, \kappa) < \frac{I_\nu(\kappa)}{I_{\nu-1}(\kappa)} < B_0(\nu, \kappa), \quad \textit{when} \quad \nu \geq 1/2 \quad (8)$$

where $B_\alpha(\nu,\kappa) = \frac{\kappa}{\delta_\alpha(\nu,\kappa)+\sqrt{\delta_\alpha(\nu,\kappa)^2+\kappa^2}}, \delta_\alpha(\nu,\kappa) = (\nu - 1/2) + \frac{\lambda}{2\sqrt{\lambda^2+\kappa^2}}, \lambda = \nu + (\alpha-1)/2$, and ν denotes the dimension and z denotes the concentration parameter. Our observation is that the gap between the upper and lower bounds of the ratio becomes tighter as the dimension grows as shown in Supplement. Even in low dimensions, the gap is less than e^{-10} for various concentration parameter values (κ). Using this fact, we simply approximate the ratio by the average over the lower and upper bounds, $\frac{I_\nu(\kappa)}{I_{\nu-1}(\kappa)} \approx \frac{B_2(\nu,\kappa)+B_0(\nu,\kappa)}{2}$. Empirically we find that this simple approximation allows us to obtain numerically stable gradients on dimensions of several thousands. Furthermore, this approximation saves us from directly computing modified Bessel function. Since the modified Bessel function of the first kind of high order is not supported yet in most deep learning frameworks, using this approximation, variational inference with high dimensional vMF distributions can utilize GPU acceleration without extra efforts on CUDA implementations of Bessel functions.

Structured compression using the radial density

Subsequently to presenting the RDP family and overcoming the optimization issues, we show the utility of the radius component in search for a more optimal model in form of model compression. Thus, using the *radial* density given in Eq.6, we can prune out output neurons on each layer depending on the contribution of each neuron measured by the learned posterior distribution. We call this scheme *row-grouping* as depicted in Fig. 2. One can also employ *column-grouping*. Or one can also prune out both input and output neurons simultaneously, using *double-grouping*, with an additional constraint that every element of the weight matrix is a product of each element of row and column matrices, i.e., $\mathbf{W}^{(l)}_{i,j} = \mathbf{w}^{(l)}_{r,i} \cdot \mathbf{w}^{(l)}_{l,j}$ to ensure $q(\mathbf{W})$ is a proper probability density (i.e., normalized), $q(\mathbf{W}) = \prod_{l=1}^{L}\prod_{r=1}^{n_l}\prod_{c=1}^{n_{l-1}} q^{(l)}_r(\mathbf{w}^{(l)}_r) q^{(l)}_c(\mathbf{w}^{(l)}_c)$, where $q^{(r)}_r(\mathbf{w}^{(l)}_r) = q^{(r)}_{c,rad}(\|\mathbf{w}^{(l)}_r\|_2) q^{(l)}_{r,dir}\left(\frac{\mathbf{w}^{(l)}_r}{\|\mathbf{w}^{(l)}_r\|_2}\right)$, and $q^{(c)}_r(\mathbf{w}^{(l)}_c) = q^{(c)}_{c,rad}(\|\mathbf{w}^{(l)}_c\|_2) q^{(l)}_{c,dir}\left(\frac{\mathbf{w}^{(l)}_c}{\|\mathbf{w}^{(l)}_c\|_2}\right)$. The prior distribution also needs to be modified according to which grouping the posterior distribution takes.

Note that for a neural network with convolutional layers, $\mathbf{W}^{(l)}$ is a convolutional filters of 4D tensor of $n_l \times n_{l-1} \times k_w \times k_h$ where n_{l-1} is the number of input channels, n_l is the number of output channels and k_w and k_h are kernel width and height. Then row-wise grouping of the filter $\mathbf{W}^{(l)}$ is row-wise grouping of 2D flattened matrix of dimension $n_l \times (n_{l-1} \cdot k_w \cdot k_h)$ which is grouped by an output channel. Column-wise grouping is to group by input channels, grouping of the flattened matrix of dimension $(n_l \cdot k_w \cdot k_h) \times n_{l-1}$.

Following Louizos, Ullrich, and Welling (2017), we use the (log of) *posterior mode* (which is the mean parameter minus the variance parameter of the log-normal posterior distribution) as a cut-off threshold to determine which output neuron needs to remain or be pruned out from the model. Recall that there are four log-normal distributions to approximate the posterior over the radius in Eq.5. Since a product of two log-normals is a log-normal with summed-up parameters from the two, we use the mode of this combined log-normal for pruning. Note that for pruning rows (or columns or both) only the two 'local' log-normal distributions in Eq.5 matter as the remainders are 'global' ones (i.e., the global ones scale up and down the local ones in the exact same way across rows/columns).

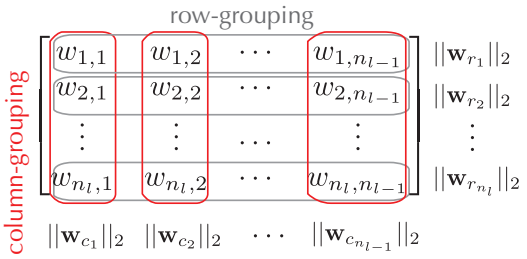

Figure 2: Column grouping (red) to prune out input neurons, depending on L2-norm of each column vector. Row grouping (grey) to prune out output neurons. Double grouping (red and grey) to prune out both input and output neurons.

4 Related Work

As mentioned in Sec. 2, many existing papers on variational BNNs assume the mean-field posterior distribution on the network weights. Recently, Louizos and Welling (2016), Sun (2017), and Zhang et al. (2018) proposed to model dependency between weights explicitly, e.g., using a matrix normal posterior with diagonal covariance matrices (Louizos and Welling 2016). There are also methods to model dependencies between weights without explicitly specifying a parametric family for the posterior (Louizos and Welling 2017; Sun 2019). However, unlike these, we capture dependency between weights through a directional component by restricting it to be on a unit sphere, which impose dependencies on weights (within a row and/or a column) directly. In addition, by coupling the concentration parameters per layer, we capture similarities among weights across rows, columns, and/or both.

The reparametrization trick is an essential tool in variational learning of BNNs. When it comes to von Mises-Fisher distribution, only the vMF reparametrization trick proposed by Davidson (2018) that implements rejection sampling (Naesseth et al. 2017) is practically applicable, due to the complexity of the sampling procedure for vMF. In our case, using the vMF posterior introduces a new challenge as scalability to high dimension is required. We improve the above approaches with a simple approximation method which makes the vMF reparametrization trick scalable up to several thousands of dimensions. Besides, the approximation yields MC gradient estimates that are corrected to be unbiased for more efficient and stable training (see Supplementary material for details).

Research on neural network compression started with non-Bayesian approaches (Hassibi and Stork 1993) and

Data	PBP	Dropout	VMG	RDP
Bos.	-2.57±.09	-2.46±.25	**-2.46±.09**	-2.60±.03
Con.	-3.16±.02	-3.04±.09	-3.01±.03	**-2.61±.02**
Ene.	-2.04±.02	-1.99±.09	**-1.06±.03**	-1.18±.03
Kin.	0.90±.01	0.95±.03	1.10±.01	**2.17±.00**
Nav.	3.73±.01	**3.80±.05**	2.46±.00	2.50±.00
Pow.	-2.84±.01	-2.80±.15	-2.82±.01	**-0.14±.01**
Pro.	-2.97±.00	-2.89±.01	-2.84±.00	**-1.34±.01**
Win.	-0.97±.01	-0.93±.06	-0.95±.01	**-0.45±.00**
Yac.	-1.63±.02	-1.55±.12	**-1.30±.02**	-2.36±.04
Yea.	-3.60±NA	-3.59±NA	-3.59±NA	**-3.51±NA**

Table 1: Average test log-likelihood on UCI regression tasks. Our method (RDP) achieves better test likelihoods (6 out of 10 datasets) than other methods.

mostly focused on non-structured pruning (Han 2015). Recently, hardware-oriented considerations have turned the research towards structured pruning for more practical speed-ups (Srinivas and Babu 2015; Li 2016; Wen 2016; Lebedev and Lempitsky 2016; Zhou, Alvarez, and Porikli 2016). Subsequently by taking into account the network weights' uncertainties, Bayesian methods have achieved an impressive compression rate. For example, Molchanov, Ashukha, and Vetrov (2017) proposed the mean-field Gaussian posterior along with sparsity inducing prior on scale parameters; and Louizos, Ullrich, and Welling (2017) proposed structuraly grouping weights through a group Horseshoe prior. In contrast, our RDP utilizes the probability density over the magnitudes of weights and prune out neurons based on captured uncertainties over the magnitudes.

5 Experiments

In this section we provide empirical evidence supporting RDP's strengths. In all experiments, we use Adam optimizer (Kingma and Ba 2014) with Pytorch default setting. In all tasks, double grouping is used. Our code is on GitHub[2].

Regression using UCI data We compare the predictive performance on regression tasks UCI dataset tested in (Gal and Ghahramani 2016; Louizos and Welling 2016) following the experimental settings from (Hernández-Lobato and Adams 2015). We split the datasets $90\%/10\%$ between training and test data.[3] To model noise variance (or precision), we use Gamma prior τ, $p(\tau) = \mathcal{G}(a_0 = 6, b_0 = 6)$ and posterior $q(\tau) = \mathcal{G}(a_1, b_1)$. We optimize for a_1, b_1 along with all the other variational parameters. The architecture used is n_{input}-50-1 except for the Protein and Year datasets where 100 hidden neurons are used. In the case of the second layer with the output dimension being one, we use fully factorized Gaussian and RDP with double grouping is only applied to the first layer. This is just enough to see improvement over mean-field based BNNs, such as Variational Inference (VI) (Graves 2011), Probabilistic Back-Propagation(PBP) (Hernández-Lobato and Adams 2015), Dropout (Gal and Ghahramani 2016). Compared to another dependency aware posterior, Variational Matrix Gaussian (VMG) (Louizos and Welling 2016), 6 out of 10 dataset, RDP shows better test log-likelihood(LL) than others. For more extensive comparison on this task, refer to Bui (2016).

[2]https://github.com/ChangYong-Oh/BayesianNeuralNetwork

[3]We follow the splitting rule given in https://github.com/yaringal/DropoutUncertaintyExps

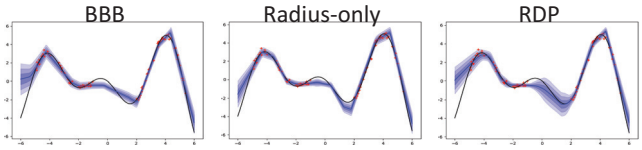

Figure 3: Predictive distributions (mean$\pm 3\times$std) from BBB, Radius-only and RDP. We chose the one with the best validation LL(on 9 equally spaced points) from 10 runs. RDP shows high uncertainty around the area where observations (red dots) are sparse.

Uncertainty quantification To illustrate the benefits of decomposing the posterior into two components in uncertainty estimation, we conduct two experiments, a 1-dim function prediction to directly visualize predictive distributions and a contextual bandit problem to indirectly test the effect of the quality of uncertainty in the downstream task.

We consider three different types of posteriors: RDP with both Radial and Directional parts, Radius-only that contains only the Radial part and BBB (Blundell et al. 2015). In the 1-dim problem we predict function $(1+x)\sin(1.2x) + 0.2\epsilon$ with 1-10-10-1 neural network architecture and ReLU activation. We observe that RDP exhibits high uncertainty in both inter/extrapolation regions without data and that RDP provides smoother predictions than others in Fig. 3.

In the *Wheel bandit* problem (Riquelme 2018), the uncertainty plays a significant role to balance between exploration and exploitation. We train networks (5-dim context and 5 actions) with 5-50-50-5 architecture and ReLU activation. Two posteriors, RDP and Radius-only (RDP without the directional component), were compared. The original context is 2-dim, but we added 3-dim dummy variables to make the problem more challenging. With 10 different random initialization, we compute the average cumulative regret (lower is better) in each case. The resulting average cumulative regret in the case of RDP is 2.4 with standard deviation 0.71 and that in the case of the Radial-only posterior was 3.6 with standard deviation 0.85.

Both experiments support that having both the directional and radial components in the posterior is helpful to capture posterior uncertainty better. Moreover, the RPD provides better-calibrated uncertainty than Radius-only and BBB.

Compression We further extend the applicability of the proposed variational family to the task of structured compression of convolutional neural network architecture. On MNIST dataset, we compress the architecture of LeNet5[4],

[4]https://github.com/BVLC/caffe/tree/master/examples/mnist

Method	Architecture	FLOPs/Params	Error
RDP	4-7-110-66	125K / 20K	1.0%
BC-GNJ	8-13-88-13	307K / 22K	1.0%
BC-GHS	5-10-76-16	169K / **15K**	1.0%
FDOO(1e5)	2-7-112-478	**119K** / 66K	1.1%
FDOO(2e5)	3-8-128-499	163K / 81K	1.0%
GL	3-12-192-500	236K / 134K	1.0%
GD	7-13-208-16	298K / 49K	1.1%
SBP	3-18-284-283	295K / 164K	**0.9%**

Table 2: The structured pruning of LeNet-5-Caffe with architecture 20-50-800-500. We benchmark our method against BC-GNJ, BC-GHS (Louizos, Ullrich, and Welling 2017), FDOO (Tang, Adhikari, and Lin 2018), Generalized Dropout(GD) (Srinivas and Babu 2015), Group Lasso(GL) (Wen 2016), Structured Bayesian Pruning(SBP) (Neklyudov 2017).

which consists of 2 convolutional layers and 2 fully-connected layers. We choose the model which has the best training cross-entropy during last 10 epochs. After training, we plot a statistics (log of mode) of radius posteriors and prune it according to clearly separated clusters as shown in Supplement. In terms of various criteria for compressed architecture, compression using RDP shows well-balanced scores such as the number of parameters, FLOPs, and loss in accuracy. We compute FLOPs for convolutional layers by $(K_w K_h C_{in}+1)(I_h+P_h-K_h+1)(I_w+P_w-K_w+1)C_{out}$ where I_h, I_w are input height and width, K_h, K_w are kernel height and width, P_w, P_h are padding height and width, and C_{in}, C_{out} are the number of input and output channels. For fully-connected layers, we compute FLOPs by $(I_{in}+1)I_{out}$, where I_{in} and I_{out} are the number of input and output neurons, respectively.

The approach achieves competitive results with the state-of-the-art methods. As given in Table 2, the RDP architecture shows particularly good compression for convolutional layers, which helps in decreasing the number of FLOPs. Direct Optimization Objective(100K) (FDOO) is only slightly less computationally heavy but at the cost of three times more parameters. Similarly, RDP comes only second to BC-GHS in terms of parameter number which though has 1.5 more FLOPs.

We also test RDP's compression capability on VGG16 and CIFAR10. Once we prune based on the radial component as in the LeNet experiment (denoted by RDP (R) in Table 3), we attempt to prune out further using the directional component (denoted by RDP (RD) in Table 3). Our hypothesis is that if many rows and columns have similar directions, removing some of them would not hurt the accuracy by much. We use the K-medoids algorithm to cluster the rows and columns per each layer (we define five clusters per layer) among the remaining rows and columns from the radial pruning. We then, remove the rows and columns that have small norms within the clusters whose total number of members are above a certain threshold[5]. Using the directional component this way, we obtain a slightly better pruning outcome than using the radial component alone. Although the RDP compression performance is slightly worse than BC-GNJ and BC-GHS, an intriguing aspect is that RDP is better at compressing convolutional layers closer to input, while BC-GNJ and BC-GHS prune almost none near the input layer and prune very heavily near the output layer.

[5]We set the threshold by kt/c with $k = 0.7$, where t is the number of neurons in a layer, c is the number of clusters in a layer.

	Architecture	Err	FLOPs Params
RDP (R)	27-57-125-122-236-244-246 -340-127-77-89-52-380-414	8.7 %	172M 3.1M
RDP (RD)	27-57-125-122-235-244-246 -335-122-72-84-47-375-409	9.1 %	170M 3M
BC (GNJ)	63-64-128-128-245-155-63 -26-24-20-14-12-11-15	8.6 %	142M 1M
BC (GHS)	51-62-125-128-228- 129-38 -13-9-6-5-6-6-20	9.0 %	122M 0.8M

Table 3: The structured pruning of VGG16. RDP show comparable performance as BC-GNJ, BC-GHS. Compared to BC-GNJ and BC-GHS, RDP compresses better convolutional layers closer to the input layer.

6 Conclusion

We proposed a new variational family to capture dependencies between weight parameters in a structured way for Bayesian Neural Networks. RDP's capability to capture the dependency between weights is empirically supported by its performance on regression tasks. Also, RDP has a natural structure for compression and it scores competitive to other methods in multiple compression performance measures. For practical implementations of variational inference with RDP, we proposed a simple approximation to the ratio of Bessel functions and a reparametrization trick for von-Mises Fisher(vMF) distribution. We expect our proposal to be useful as there is a wide range of applications where variational inference with high dimensional vMF distributions could be useful.

Acknowledgments

We are grateful to the Max Planck Society for its support. M. Park also thanks to the Gibs Schüle Foundation and the Institutional Strategy of the University of Tübingen (ZUK63).

References

Banerjee, A. e. a. 2005. Clustering on the unit hypersphere using von mises-fisher distributions. *JMLR* 6(Sep):1345–1382.

Blundell, C.; Cornebise, J.; Kavukcuoglu, K.; and Wierstra, D. 2015. Weight uncertainty in neural networks. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning-Volume 37*, 1613–1622. JMLR. org.

Bui, T. e. a. 2016. Deep gaussian processes for regression using approximate expectation propagation. In *ICML*, 1472–1481.

Davidson, T. R. e. a. 2018. Hyperspherical variational auto-encoders. *arXiv preprint arXiv:1804.00891*.

Dayan, P., and Hinton, G. E. 1996. Varieties of helmholtz machine. *Neural Networks* 9(8):1385–1403.

Figurnov, M.; Mohamed, S.; and Mnih, A. 2018. Implicit reparameterization gradients. In *Advances in Neural Information Processing Systems*, 441–452.

Gal, Y., and Ghahramani, Z. 2016. Dropout as a bayesian approximation. In *ICML*, 1050–1059.

Ghosh, S. e. a. 2018. Structured variational learning of Bayesian neural networks with horseshoe priors. In *ICML*, 1744–1753.

Graves, A. 2011. Practical variational inference for neural networks. In *NIPS*, 2348–2356.

Han, S. e. a. 2015. Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv:1510.00149*.

Hassibi, B., and Stork, D. G. 1993. Second order derivatives for network pruning: Optimal brain surgeon. In *NIPS*, 164–171.

Hernández-Lobato, J. M., and Adams, R. 2015. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *ICML*, 1861–1869.

Hoffman, M. D. e. a. 2013. Stochastic variational inference. *JMLR* 14(1):1303–1347.

Jang, E.; Gu, S.; and Poole, B. 2016. Categorical reparameterization with gumbel-softmax. *arXiv:1611.01144*.

Kingma, D. P., and Ba, J. 2014. A method for stochastic optimization. *arXiv:1412.6980*.

Kingma, D. P., and Welling, M. 2013. Auto-encoding variational bayes. *arXiv:1312.6114*.

Kingma, D. P.; Salimans, T.; and Welling, M. 2015. Variational dropout and the local reparameterization trick. In *NIPS*, 2575–2583.

Lakshminarayanan, B.; Pritzel, A.; and Blundell, C. 2017. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, 6402–6413.

Lebedev, V., and Lempitsky, V. 2016. Fast convnets using group-wise brain damage. *CVPR*.

Li, H. e. a. 2016. Pruning filters for efficient convnets. *arXiv:1608.08710*.

Louizos, C., and Welling, M. 2016. Structured and efficient variational deep learning with matrix gaussian posteriors. In *ICML*, 1708–1716.

Louizos, C., and Welling, M. 2017. Multiplicative normalizing flows for variational bayesian neural networks. In *International Conference on Machine Learning*, 2218–2227.

Louizos, C.; Ullrich, K.; and Welling, M. 2017. Bayesian compression for deep learning. In *NIPS*, 3288–3298.

MacKay, D. J. 1995. Probable networks and plausible predictions–a review of practical bayesian methods for supervised neural networks. *Network: computation in neural systems* 6(3):469–505.

Maddison, C. J. e. a. 2016. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv:1611.00712*.

Mardia, K. V., and Jupp, P. E. 2009. *Directional statistics*, volume 494. John Wiley & Sons.

Molchanov, D.; Ashukha, A.; and Vetrov, D. 2017. Variational dropout sparsifies deep neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, 2498–2507. JMLR. org.

Naesseth, C.; Ruiz, F.; Linderman, S.; and Blei, D. 2017. Reparameterization gradients through acceptance-rejection sampling algorithms. In *International Conference on Artificial Intelligence and Statistics*.

Neal, R. M. 2012. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media.

Neklyudov, K. e. a. 2017. Structured bayesian pruning via log-normal multiplicative noise. In *NIPS*, 6775–6784.

Neville, S. E. e. a. 2014. Mean field variational bayes for continuous sparse signal shrinkage: pitfalls and remedies. *Electronic Journal of Statistics* 8(1):1113–1151.

Riquelme, C. e. a. 2018. Deep bayesian bandits showdown. In *ICLR*.

Ritter, H.; Botev, A.; and Barber, D. 2018. A scalable laplace approximation for neural networks. In *ICLR*.

Ruiz-Antolín, D., and Segura, J. 2016. A new type of sharp bounds for ratios of modified bessel functions. *JMAA* 443(2):1232–1246.

Slingo, J., and Palmer, T. 2011. Uncertainty in weather and climate prediction. *Phil. Trans. R. Soc. A* 369(1956):4751–4767.

Srinivas, S., and Babu, R. V. 2015. Data-free parameter pruning for deep neural networks. *arXiv:1507.06149*.

Sun, S. e. a. 2017. Learning structured weight uncertainty in bayesian neural networks. In *AISTATS*, 1283–1292.

Sun, S. e. a. 2019. Functional variational bayesian neural networks. *arXiv:1903.05779*.

Tang, R.; Adhikari, A.; and Lin, J. 2018. Flops as a direct optimization objective for learning sparse neural networks. *arXiv:1811.03060*.

Tucker, G. e. a. 2017. Low-variance, unbiased gradient estimates for discrete latent variable models. In *NIPS*, 2627–2636.

Wen, W. e. a. 2016. Learning structured sparsity in deep neural networks. In *NIPS*, 2074–2082.

Zhang, G.; Sun, S.; Duvenaud, D.; and Grosse, R. 2018. Noisy natural gradient as variational inference. In *International Conference on Machine Learning*, 5847–5856.

Zhou, H.; Alvarez, J. M.; and Porikli, F. 2016. Less is more: Towards compact cnns. In *ECCV*, 662–677. Springer.