Radial Bayesian Neural Networks: Beyond Discrete Support In Large-Scale Bayesian Deep Learning

Sebastian Farquhar[†] **Michael A. Osborne*** **Yarin Gal**[†]

Abstract

We propose Radial Bayesian Neural Networks (BNNs): a variational approximate posterior for BNNs which scales well to large models while maintaining a distribution over weight-space with full support. Other scalable Bayesian deep learning methods, like MC dropout or deep ensembles, have discrete support—they assign zero probability to almost all of the weight-space. Unlike these discrete support methods, Radial BNNs' full support makes them suitable for use as a prior for sequential inference. In addition, they solve the conceptual challenges with the *a priori* implausibility of weight distributions with discrete support. The Radial BNN is motivated by avoiding a sampling problem in 'mean-field' variational inference (MFVI) caused by the so-called 'soap-bubble' pathology of multivariate Gaussians. We show that, unlike MFVI, Radial BNNs are robust to hyperparameters and can be efficiently applied to a challenging real-world medical application without needing ad-hoc tweaks and intensive tuning. In fact, in this setting Radial BNNs out-perform discrete-support methods like MC dropout. Lastly, by using Radial BNNs as a theoretically principled, robust alternative to MFVI we make significant strides in a Bayesian continual learning evaluation.

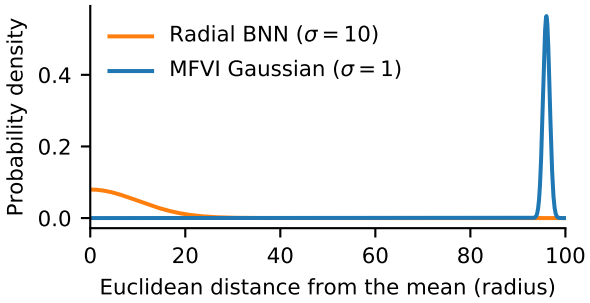

Figure 1: MFVI uses a multivariate Gaussian approximate posterior whose probability mass is tightly clustered at a fixed radius from the mean depending on the number of dimensions—the 'soap-bubble'. In our Radial BNN, samples from the approximate posterior are more reflective of the mean. This helps training by reducing gradient variance. (Plotted p.d.f. is based on dimensionality of a 3x3 conv layer with 64 channels.)

1 INTRODUCTION

The most effective scalable methods for Bayesian deep learning have a significant shortcoming: they learn an approximate posterior distribution that has discrete support over the weight-space—the probability assigned to almost all possible weights is zero. This is true of methods like MC dropout [Gal and Ghahramani, 2015], stochastic gradient Markov Chain Monte Carlo (MCMC) [Welling and Teh, 2011], or deep ensembles [Lakshminarayanan et al., 2016]. This is implausible *a priori*: we are fairly certain the true posterior over the weights ought not be zero almost everywhere. But it is also unhelpful because it makes these distributions unsuitable as a data-dependent prior in sequential inference tasks—we are in the dark about what the prior probability of almost all sets of weights should be.

Some variational inference methods *do* learn approximate posteriors with full support over the weight-space. 'Mean-field' variational inference (which assumes that network weights are independent of each other) is fast and has linear time complexity in the number of parameters [Hinton and van Camp, 1993, Graves, 2011, Blundell et al., 2015]. Unfortunately, MFVI struggles in practice for tasks larger than roughly the scale of MNIST and is highly sensitive to hyperparameters [Wu et al., 2019]. Tuning hyperparameters becomes a major barrier to using MFVI for larger models where each

† OATML Research Group, Department of Computer Science, University of Oxford.
* Department of Engineering, University of Oxford

iteration could take days to optimize. To make MFVI work, researchers often resort to ad-hoc tweaks to the loss or optimization process which side-step the variational inference arguments that motivate the approach in the first place![1] Other research has tried to relax MFVI's independence assumption by introducing expensive techniques which are only tractable for small networks with hundreds or thousands of parameters and low-dimensional problems (e.g., MNIST) [Louizos and Welling, 2016, Sun et al., 2017, 2019, Oh et al., 2019, Wu et al., 2019]. **What is missing is a robust and scalable inference procedure for BNNs that maintains full support.**

In this paper we identify a sampling problem at the heart of MFVI's failures—typical samples from the multivariate Gaussian approximate posterior used in MFVI are unrepresentative of the most-probable weights, and this problem gets worse for larger networks [Bishop, 2006]. Probability mass in a multivariate Gaussian is clustered in a narrow 'soap-bubble' far from the mean (see Figure 1). This leads to exploding gradient variance unless training is stopped before the weight variance grows too large, and prevents MFVI from actually fitting to the loss. We demonstrate this in §5.

Motivated by this, we propose an alternative approximate posterior distribution which does not have this 'soap-bubble' pathology. The Radial BNN defines a simple approximate posterior distribution in a hyperspherical space corresponding to each layer, and then transforms this distribution into the coordinate system of the weights. The typical samples from this distribution tend to come from areas of high probability density. We show that the Radial BNN can be sampled efficiently in weight-space, without needing to explicitly transform samples generated in a hyperspherical space, and derive an expression for the loss that makes training as fast and as easy to implement as MFVI.

We establish the robustness and performance of Radial BNNs using a Bayesian medical imaging task identifying diabetic retinopathy in 'fundus' eye images [Leibig et al., 2017] in §4.1 (see Figure 2). We show that Radial BNNs are much more robust to hyperparameters than MFVI and that Radial BNNs outperform the current state-of-the-art Monte-Carlo (MC) dropout and deep ensembles on this task. This involves us scaling Radial BNNs to models with ∼15M parameters accepting inputs with ∼230,000 dimensions.

In addition, because our variational inference approach produces a posterior with *full support* over the weight-space, we show that we can use our Radial BNN posterior as a prior for further inference. We demonstrate this in §4.2 using a continual learning setting, based on Kirkpatrick et al. [2017] and Nguyen et al. [2018], in which we use a sequence of approximate posteriors as a prior while learning a next task without forgetting what was learned in earlier tasks. While we do not solve continual learning, we use the problem setting to demonstrate the potential of our method to find rich data-dependent priors.

[1] We discuss this in detail in §4.1.2.

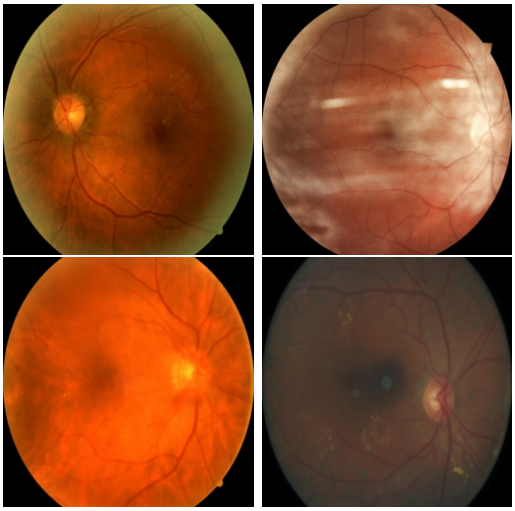

Figure 2: Examples from retinopathy dataset. Top L: healthy eye. Top R: healthy eye with camera artefacts. Bottom: diseased eyes. The chance that bad images cause misdiagnosis makes uncertainty-aware models vital. Input dimension 334x bigger than MNIST.

2 PRIOR WORK

Instead of point estimates, Bayesian neural networks (BNNs) place a parameterized distribution over each weight in a neural network [MacKay, 1992, Neal, 1995]. Many efficient approximations have been proposed to estimate the posterior distribution over those weights including mean-field variational inference [Hinton and van Camp, 1993, Graves, 2011, Blundell et al., 2015], Monte Carlo (MC) dropout [Gal and Ghahramani, 2015], stochastic gradient Markov Chain MC [Welling and Teh, 2011] and others. Deep ensembles [Lakshminarayanan et al., 2016] have also been proposed as a way to learn a distribution over the weights (although the connection to the posterior remains unclear).

Unfortunately, the robust methods that scale to large models and datasets which have been discovered so far do not learn a posterior with full support over the weight-space. That is, for the scalable methods, the probability density estimated almost everywhere is zero—this is almost certainly not the true posterior, and is unsuitable for use as a prior.

Mean-field variational inference offers a fully supported

distribution over the weights. It sets an approximate posterior distribution over each weight in the network—an independent Gaussian. It then optimizes a lower-bound on the marginal likelihood which tries to find the approximate posterior with the smallest KL-divergence to the true posterior. This loss can be interpreted as balancing predictive accuracy on the data, the entropy of the posterior, and the cross-entropy between the prior and posterior. In the common case of a unit multivariate Gaussian prior and an approximate posterior $\mathcal{N}(\mu_i, \sigma_i)$ over the weights w_i, the evidence lower bound (ELBO) objective is:

$$\mathcal{L}_{MFVI} = \overbrace{\sum_i \frac{1}{2}\left[\sigma_i^2 + \mu_i^2\right]}^{\text{prior cross-entropy}} - \overbrace{\sum_i \log[\sigma_i]}^{\text{posterior entropy}} - \overbrace{\mathbb{E}_{\mathbf{w}\sim q_\theta(\mathbf{w})}\left[\log p(\mathbf{y}|\mathbf{w}, \mathbf{X})\right]}^{\text{data likelihood}}. \quad (1)$$

In practice, training BNNs with MFVI is difficult. For example, Wu et al. [2019] argue that it is sensitive to initialization and priors. Others worry that the mean-field approximation is too constraining. Louizos and Welling [2016], Sun et al. [2017, 2019] and Oh et al. [2019] have all introduced richer variational distributions which permit correlations between weights to be learned by the BNN. Unfortunately, these methods are considerably more computationally expensive than MFVI and have only been demonstrated on problems *at MNIST scale or below*. Wu et al. [2019] instead see the variance of ELBO estimates as the problem and introduce a deterministic alternative. We agree that this is a crucial problem, but offer a simpler and cheaper alternative solution which does not require extra assumptions about the distribution of activations. Note that Osawa et al. [2019] present scalable inference for MFVI using variational online Gauss-Newton methods [Khan et al., 2018]. However, this method relies on several significant approximations to make estimates of the Hessian tractable and the performance of the method lags significantly behind deep ensembles, which we compare to.

We note that there is a superficial similarity between our method and Oh et al. [2019], insofar as they also make use of a hyperspherical coordinate system for variational inference. However, they use this coordinate system over each row in their weight matrix, rather than the whole layer, and introduce an expensive posterior distribution (von Mises-Fisher) to explicitly model weight correlations within rows, whereas we do *not* seek to learn any correlations between parameters in the hyperspherical space. That is, their method uses a different technique to solve a different problem.

3 METHOD

A well-known property of multivariate Gaussians in high dimensions is that the probability mass concentrates in a 'soap-bubble'—a narrow shell at a radius determined by the variance and the number of dimensions (e.g., [Bishop, 2006]).[2] This has the consequence that almost all samples from the distribution are very distant from the mean. All else equal, we might expect this to lead to predictions and losses which are less correlated with each other than if the samples were near to each other in weight-space. Moreover, the distance of typical samples from the approximate posterior over each layer from the mean is $\sim \sigma\sqrt{D}$ for standard deviation parameter σ and the number of parameters in the layer D, in the domain of large D typically found in modern neural networks.[3] We anticipate (and demonstrate in §5) that the distance between samples from the MFVI approximate posterior makes the gradient estimator of the log-likelihood term of the loss in Equation (1) have a large variance, which makes optimization difficult.

3.1 The Radial BNN Posterior

The pathology arises because, for large D, the probability density function over the radius from the mean is sharply peaked at a large distance from the mean (see Figure 1). Therefore, we pick a probability distribution which cannot have this property. We can easily write down a probability density function which cannot have a soap-bubble by explicitly parameterizing the radius from the mean. The hyperspherical coordinate system suits our needs: the first dimension is the radius and the remaining dimensions are angles. We pick the simplest practical distribution in hyperspherical coordinates with no soap bubble:

- In the radial dimension: $r = |\tilde{r}|$ for $\tilde{r} \sim \mathcal{N}(0, 1)$.
- In the angular dimensions: uniform distribution over the hypersphere—all directions equally likely.

A critical property is that it is easy to sample this distribution *in the weight-space coordinate system* without needing to sample in the hyperspherical coordinate system and then explicitly transform the samples, which

[2] We refer readers to the Appendix A for more detail on this phenomenon. Intuitively, the issue arises because the space expands with the polynomial r^D in the radius r in D dimensions, while the p.d.f. of the Gaussian falls exponentially. At the origin, the polynomial term is small, at infinity the exponential term is small, and almost all the probability mass lies in a narrow band in between.

[3] To calculate the distance of typical samples from the mean we imagine an isotropic posterior. Our posteriors are not isotropic, but the pattern is similar.

Method	Architecture	# Params	Epoch Train Time (m)	ROC-AUC for different percent data referred to experts 0%	10%	20%	30%
MC-dropout	[Leibig et al., 2017]	∼21M	-	92.7±0.3%	93.8±0.3%	94.7±0.3%	95.6±0.3%
MC-dropout	VGG-16	∼15M	5.6	93.0±0.04%	94.1±0.05%	94.5±0.05%	95.1±0.07%
MFVI	VGG-16*	∼15M	16.0	63.6±0.13%	63.5±0.09%	63.5±0.09%	62.6±0.10%
MFVI w/ tweaks	VGG-16*	∼15M	16.0	93.9±0.04%	94.4±0.05%	95.4±0.04%	96.4±0.05%
Radial BNN	VGG-16*	∼15M	16.2	**94.3±0.04%**	**95.3±0.06%**	**96.1±0.06%**	**96.8±0.04%**
Deep Ensemble	3xVGG-16	∼45M	16.8†	93.9±0.04%	96.0±0.05%	96.6±0.04%	97.2±0.04%
Radial Ensemble	3xVGG-16*	∼45M	48.6†	**94.5±0.05%**	**97.9±0.04%**	**98.0±0.03%**	**98.1±0.03%**

Table 1: Diabetic Retinopathy Prescreening: Our Radial BNN outperforms SOTA MC-dropout and is able to scale to model sizes that MFVI cannot handle without ad-hoc tweaks (see §4.1.2). Even with tweaks, Radial BNN still out-performs. Deep Ensembles outperform a single Radial BNN at estimating uncertainty, but are worse than an ensemble of Radial BNNs with the same number of parameters. ± indicates bootstrapped standard error from 100 resamples of the test data. VGG-16* model has fewer channels so that # of parameters is the same as non-Bayesian model. † 3x single model train time. Could be in parallel.

would be expensive. Instead of sampling the posterior distribution directly, we use the local reparameterization trick Rezende et al. [2014], Kingma et al. [2014], and sample the noise distribution instead. This is similar to Graves [2011], Blundell et al. [2015] who sample their weights according to:

$$\mathbf{w} \coloneqq \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}_{\text{MFVI}} \tag{2}$$

where $\boldsymbol{\epsilon}_{\text{MFVI}} \sim \mathcal{N}(0, \mathbf{I})$. In order to sample from the Radial BNN posterior we make a small modification:

$$\mathbf{w}_{\text{radial}} \coloneqq \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \frac{\boldsymbol{\epsilon}_{\text{MFVI}}}{\|\boldsymbol{\epsilon}_{\text{MFVI}}\|} \cdot r \tag{3}$$

which works because dividing a multi-variate Gaussian by its norm provides samples from a direction uniformly selected from the unit hypersphere [Muller, 1959, Marsaglia, 1972]. As a result, sampling from our posterior is nearly as cheap as sampling from the MFVI posterior. The only extra steps are to normalize the noise, and multiply by a scalar Gaussian.

3.2 Evaluating the Objective

To use the our approximate posterior for variational inference we must be able to estimate the ELBO loss. The Radial BNN posterior does not change how the expected log-likelihood is estimated, using mini-batches of datapoints and MC integration. However, the KL divergence between the approximate posterior and prior needs a new estimator. This divergence splits into two terms:

$$\begin{aligned} KL\big(q(\mathbf{w}) \parallel p(\mathbf{w})\big) &= \int q(\mathbf{w}) \log[q(\mathbf{w})] d\mathbf{w} \\ &\quad - \int q(\mathbf{w}) \log\big[p(\mathbf{w})\big] d\mathbf{w} \\ &= \mathcal{L}_{\text{entropy}} - \mathcal{L}_{\text{cross-entropy}}. \end{aligned} \tag{4}$$

We estimate the cross-entropy term using MC integration as well, just by taking samples from the posterior and averaging their log probability under the prior. We find that this is low-variance in practice, and is often done for MFVI as well [Blundell et al., 2015].

We can evaluate the entropy of the posterior analytically. We derive the entropy term in Appendix B:

$$\mathcal{L}_{\text{entropy}} = -\sum_i \log[\sigma_i] + \text{const.} \tag{5}$$

where i sums over the weights. This is, up to a constant, the same as when using an ordinary multivariate Gaussian in MFVI. (For sake of completeness, we also derive the constant terms in the Appendix.) In Appendix C, we also provide a derivation of the cross-entropy loss term in the case where the prior is a Radial BNN. This is useful in continual learning (see §4.2) where we use the posterior from training one model as a prior when training another.

3.3 Computational Complexity

Training Radial BNNs has the same computational complexity as MFVI—$\mathcal{O}(D)$, where D is the number of weights in the model. In contrast, recent non-mean-field extensions to VI like Louizos and Welling [2016] and Sun et al. [2017] have higher time complexities. For example, Louizos and Welling [2016] uses a pseudo-data approximation which reduces their complexity to $\mathcal{O}(D + M^3)$ where M is a pseudo-data count. But even for MNIST, they use M up to 150 which becomes quite significant (this is their largest experiment). Sun et al. [2017] have the same complexity as Louizos and Welling [2016], depending on similar approximations and consider a maximum input dimension of only 16—over 16,000x smaller than the input dimension of the task we address in §4.1. In practice, the comparison between our Radial BNNs and MFVI is even more favorable. Despite identical time complexity, we found that because our method was much more robust to hyperparameters than MFVI, the most accurate hy-

perparameter configuration we discovered in §4.1.3 for Radial BNNs trained roughly four times faster than the most accurate MFVI configuration.

4 EXPERIMENTS

Our work is focused on large datasets and big models, which is where the most exciting application for deep learning are. That is where complicated variational inference methods that try to learn weight covariances become intractable, and where the 'soap-bubble' pathology emerges.

We address this head-on in §4.1. We show that on a large-scale diabetic retinopathy diagnosis image classification task: our radial posterior is *more accurate*, has *better calibrated uncertainty*, and is *more robust* to hyperparameters than MFVI with a multivariate Gaussian and therefore requires significantly fewer iterations and less experimenter time. In this setting, we have ∼260,000 input dimensions and use a model with ∼15M parameters. This is orders of magnitude larger than most other VI work, has been heavily influenced by the experimental settings used to evaluate performance on UCI datasets by Hernández-Lobato and Adams [2015] with between 4 and 16 input dimensions and using fewer than 2000 parameters.[4]

In §4.2, we show that we can use the posterior from variational inference with Radial BNNs as a prior when learning future tasks. We demonstrate this using a continual learning problem [Kirkpatrick et al., 2017] on FashionMNIST [Xiao et al., 2017]. We show significantly improved performance relative to the MFVI-based Variational Continual Learning (VCL) introduced by Nguyen et al. [2018].

4.1 Diabetic Retinopathy Prescreening

We perform classification on a dataset of 'fundus' images taken of the back of retinas in order to diagnose diabetic retinopathy [Kaggle, 2015] building on Leibig et al. [2017] and Filos et al. [2019]. Diabetic retinopathy is graded in five stages, where 0 is healthy and 4 is the worst. Following Leibig et al. [2017], we distinguish the healthy (classes 0 and 1) from those that require medical observation and attention (2, 3, and 4). Images (512x512) include left and right eyes separately, which are not considered as a pair by the models, and come from two different camera technologies in many different physical locations. We use model uncertainty to identify badly taken or confusing images and refer these patients to experts for more detailed examination.

[4]Radial BNNs, like MFVI, do not match the performance of some of the more expensive methods on the UCI datasets. We would not expect it to—our method is specifically designed for models with high-dimensional weight-space, not for the artificial constraints of the experimental settings used on the UCI evaluations. See Appendix E for details.

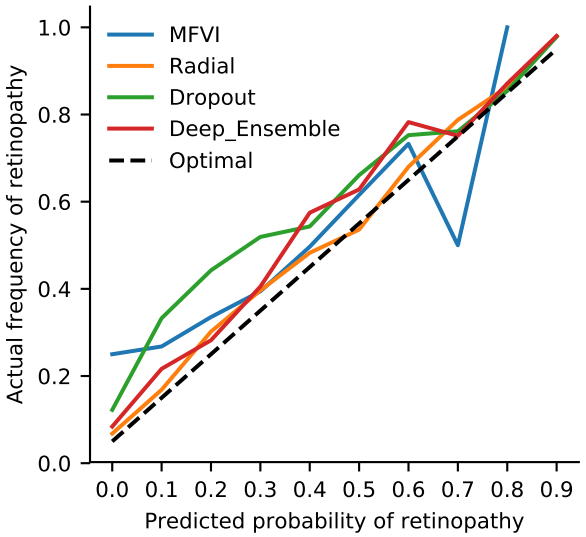

Figure 3: Radial BNN is almost perfectly calibrated, compared with MC dropout and deep ensembles (overconfident) and ordinary MFVI without ad-hoc tweaks which is not well calibrated. X-axis labels are the lower-bound of each range (e.g., 0.0 is 0.0-0.1).

4.1.1 Performance and Calibration

In Table 1 we compare the classification area under the curve (AUC) of the receiver operating characteristic of predicted classes (higher is better).[5] We consider the model performance under different thresholds for referring data to experts. At 0%, the model makes predictions about all data. At 30%, the 30% of images about which the model is least confident are referred to experts and do not get scored for the model—the AUC should therefore become higher if the uncertainties are well-calibrated. We show that our Radial BNN outperforms MFVI by a wide margin, and even outperforms MC dropout, a highly effective approximate VI method. While the deep ensemble is better at estimating uncertainty than a single Radial BNN, it has three times as many parameters. An ensemble of Radial BNNs outperforms deep ensembles at all levels of uncertainty. The model hyperparameters were all selected individually by Bayesian optimization using ten runs. Full hyperparameters and search strategy, preprocessing, and architecture are provided in Appendix D.1.

We include both the original MC dropout results from Leibig et al. [2017] as well as our reimplementation using the same model architecture as our Radial BNN model. The only difference between the MC dropout and Radial BNN/MFVI architectures is that we use

[5]We use AUC because classes are unbalanced (mostly healthy): accuracy gives distorted picture of performance.

more channels for MC dropout so that the number of parameters is the same in all models. We estimate standard error of the AUC using bootstrapping.

4.1.2 MFVI Tweaks

In some cases, researchers have been able to get MFVI to work by applying various ad-hoc tweaks to the training process. Here, we evaluate the performance of these tweaks and in §5 we explain how the success of these tweaks aligns with our hypothesis that MFVI suffers from a sampling problem which Radial BNNs fix.

One approach to making MFVI work is to pre-train the means of the model using the ordinary log-likelihood loss and to switch partway through training to the ELBO loss, initializing the weight variances at this point with a very small value approximating a deterministic neural network. (E.g., [Nguyen et al., 2018] who initialize with a variance of 10^{-6} after pre-training the means.) If one trains to convergence, the weight variances will tend to grow bigger than their tiny initialization, which destroys model performance in MFVI, so one must also employ early stopping.[6] We find that if we perform all these ad-hoc tweaks we are indeed able to get acceptable performance on our diabetic retinopathy dataset (see Table 1), though still worse than Radial BNNs. But the tweaks mean that the learned distribution can certainly not be regarded as an approximate posterior based on optimizing the ELBO. Moreover, the tweaks amount to approximating a deterministic network.

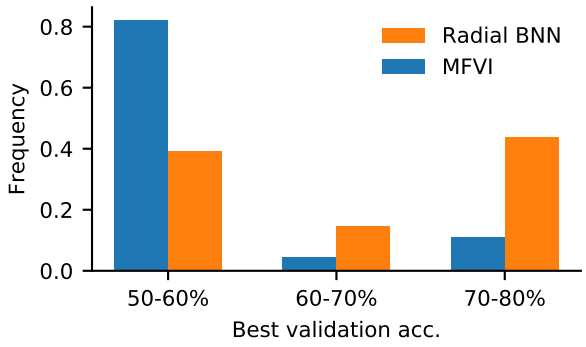

Figure 4: Our Radial BNN posterior is much more robust to hyperparameters on a downsampled version of the retinopathy dataset. Over 80% of configurations for the MFVI baseline learned almost nothing. 4x more Radial BNNs than MFVIs had good accuracies.

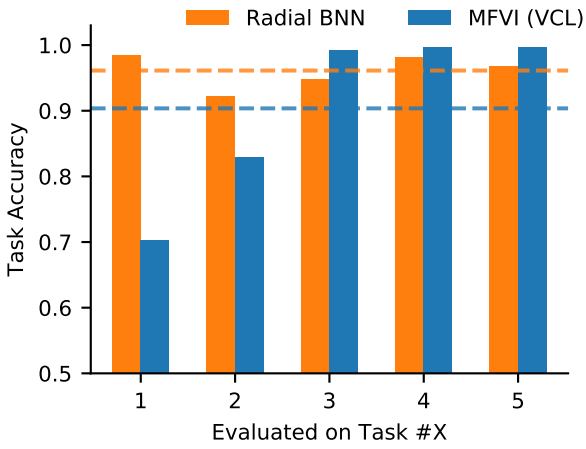

Figure 5: The five FashionMNIST tasks arrive in series. The model only sees data from the newest task and uses the posterior from the previous task to remember *all* earlier tasks. Here we show performance of the final model, trained on all 5 tasks in sequence. MFVI (as used in VCL) gradually forgets tasks—the final model's accuracy is worse the older the tasks get. Our Radial BNN preserves information and is still good at the first task. Average acc. shown by the dotted line.

4.1.3 Robustness

We also found that the radial posterior was much more robust to hyperparameter variation (Figure 4). We assess robustness on a downsampled version of the diabetic retiopathy dataset (256x256) using a smaller model with a similar architecture to VGG-16, but which trained to convergence in about a tenth the time and had only ∼1.3M parameters. We randomly selected 86 different runs from plausible optimizer, learning rate, learning rate decay, batch size, number of variational samples per forward pass, and initial variance. 82% of hyperparameters tried for the MFVI baseline resulted in barely any improvement over randomly guessing, compared with 39% for the radial posterior. 44% of configurations for our radial posterior reached good AUCs, compared with only 11% for MFVI. This is despite the fact that we did allow models to pre-train the means using an NLL loss for one epoch before begining ELBO training, a common tweak to improve MFVI.

4.2 Continual Learning

Continual learning is a problem setting where a sequence of tasks must be learned separately while a single model is carried from one task to the next

[6] Other authors, e.g., Fortunato et al. [2018], achieve a similar result just by ignoring the KL to the prior in the loss so that the weight variances tend to shrink to overfit the training data.

but all data are discarded [Kirkpatrick et al., 2017]. This is hard because neural networks tend to exhibit 'catastrophic forgetting' and lose performance on previous tasks. A number of authors have proposed prior-focused Bayesian approaches to continual learning in which the posterior at the end of learning a task becomes a prior when learning the next task [Kirkpatrick et al., 2017, Zenke et al., 2017, Chaudhry et al., 2018, Nguyen et al., 2018, Farquhar and Gal, 2018b, Ritter et al., 2018]. In the case of exact Bayesian updating, this ought to balance the information learned from the datasets of each tasks. But for approximate methods we have no such guarantee. The better the posterior approximation, the better we might expect such prior-focused Bayesian approaches to work.

Variational Continual Learning (VCL), by Nguyen et al. [2018], applies MFVI to learning the posterior. Here, we use VCL as a problem setting to evaluate the quality of the posterior. Note that we do not aim to solve the continual learning problem, but rather to demonstrate the improvement offered by Radial BNNs to the posterior approximation. A good posterior estimate should work as an effective prior and prevent forgetting. This setting is particularly relevant to variational inference, as other methods for estimating uncertainty in neural networks (such as Monte-Carlo dropout [Gal and Ghahramani, 2015] or ensembles [Lakshminarayanan et al., 2016]) cannot be straightforwardly used during training as a prior because the posteriors they learn put zero probability for almost all weight values.

We consider a sequence of five tasks known as Split FashionMNIST [Nguyen et al., 2018, Farquhar and Gal, 2018a]. FashionMNIST is a dataset of images of items of clothing or attire (shoes, t-shirts, handbags etc.) [Xiao et al., 2017]. The first task is to classify the first two classes of FashionMNIST, then the next two etc. We examine a multi-headed model [Chaudhry et al., 2018, Farquhar and Gal, 2018a] in order to evaluate the quality of the posterior, although this is a limited version of continual learning. The models are BNNs with four hidden layers with 200 weights in each (∼250k parameters). We perform an extensive grid search over hyperparameters. Full hyperparameters and a more thorough description of the experimental settings, as well as results for the single-headed continual learning setting, are in Appendix D.2.

The Radial BNN approximate posterior acts as a better prior, showing that it learns the true posterior better (Figure 5). Radial BNNs maintain good accuracy on old tasks even after training on all five tasks. In contrast, the MFVI posterior gets increasingly less accurate on old tasks as training progresses. The MFVI posterior approximation is not close enough to the true posterior to carry the right information to the next task.

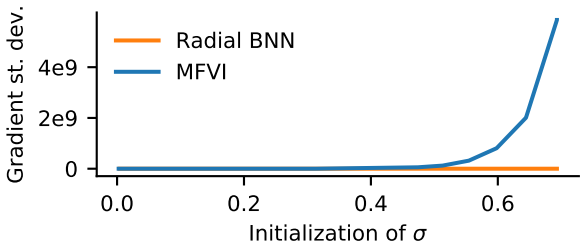

Figure 6: The variance of gradient estimates in the standard MFVI posterior explodes as the weight variance parameter grows.

5 ANALYSING RADIAL BNNs

Why is it that Radial BNNs offer improved performance relative to MFVI? In §3 we observed that the multivariate Gaussian distribution typically used in MFVI features a 'soap-bubble'—almost all of the probability mass is clustered at a radius proportional to $\sigma\sqrt{D}$ from the mean in the large D limit (illustrated in Figure 1). This has two consequences in larger models. First, unless the weight variances are very small, a typical sample from the posterior has a high L_2 distance from the means. Second, because the mass is distributed uniformly over the hypersphere that the soap-bubble clusters around, each sample from the multivariate Gaussian has a high expected L_2 distance from every other sample (similarly proportional to $\sigma\sqrt{D}$). This means that as σ and D grow, samples from the posterior are very different from each other, which we might expect to result in high gradient variance.

In contrast, in Radial BNNs the expected distance between samples from the posterior is independent of D for the dimensionality typical of neural networks. The expected L_2 distance between samples from a unit hypersphere rapidly tends to $\sqrt{2}$ as the number of dimensions increases. Since the radial dimension is also independent of D, the expected L_2 distance between samples from the Radial BNN is independent of D. This means that, even in large networks, samples from the Radial BNN will tend to be more representative of each other. As a result, we might expect that the gradient variance is less of a problem.

Indeed, this is exactly what we find. In Figure 6 we show that for the standard MFVI posterior in a 3x3 conv layer with 512 channels, the variance of initial gradients explodes after the weight standard deviation exceeds roughly 0.3. This matters because, for MFVI with a unit Gaussian prior, the KL-divergence term of the loss is minimized w.r.t. σ_i at 1—well within the region where gradient noise has exploded.

We can track this effect as it kicks in during training. In Figure 7 we show a sample training run on the down-sampled version of the diabetic retinopathy dataset

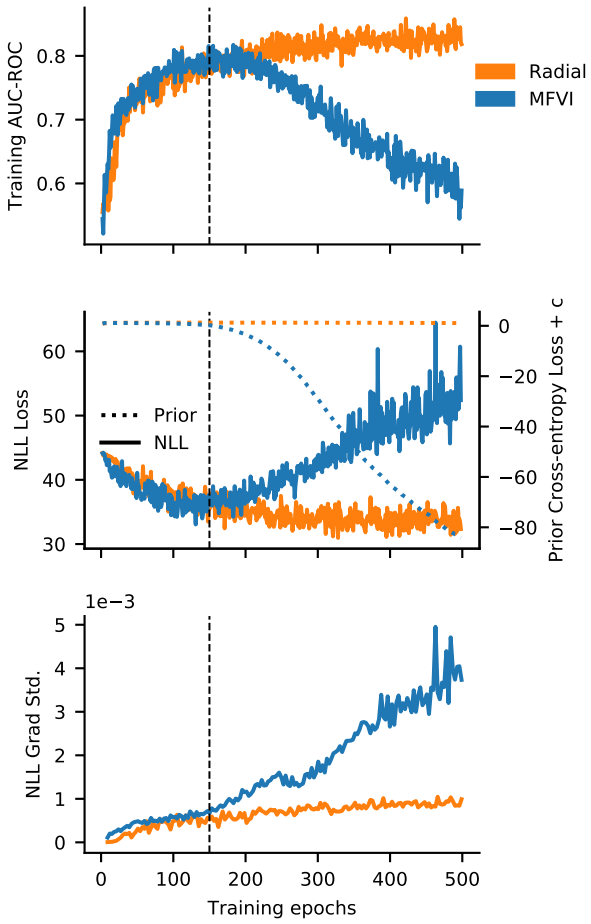

Figure 7: We can track the deterioration of the MFVI training dynamics. **Top**: After ∼150 epochs (dashed line) *training* set performance degrades for MFVI while Radial continues to improve. **Middle**: for MFVI the NLL term of the loss *increases* during training, but the prior cross-entropy term falls faster so the overall loss continues to fall. **Bottom**: The standard deviation of the NLL gradient estimator grows sharply for MFVI after about 150 epochs. This coincides with the moment where the loss is optimized by minimizing the prior cross-entropy and sacrificing the NLL.

using an MFVI and Radial BNN with the same hyperparameters. Pathologically, the *training* accuracy falls for MFVI after about 150 epochs (top graph). The critical moment corresponds to the point where the training process begins to optimize the prior cross-entropy term of the loss, sacrificing the negative log-likelihood term (middle graph). We can further show that this corresponds to the point where the standard deviation of the negative log-likelihood term of the gradient begins to sharply increase for MFVI. Meanwhile the prior cross-entropy term is computed analytically so its variance does not grow as the values of σ increase during training from their tiny initializations.

That is, MFVI fails because the high variance of the NLL term of the loss causes the optimizer to improve the cross-entropy term at the expense of the NLL term. For Radial BNNs, however, the NLL gradient variance stays low throughout training.

In Appendix F we offer further analysis of the failure of MFVI demonstrating that a biased but low-variance estimator of the gradient (using a truncated posterior approximation) improves training in MFVI.

5.1 Avoiding the Pathology in MFVI

Based on this analysis, we can see why Radial BNNs fix a sampling problem in MFVI. But this also helps explain why the ad-hoc tweaks which researchers have been using for MFVI have been successful. These tweaks chiefly serve to keep the weight variance low. Researchers initialize with small variances [Blundell et al., 2015, Fortunato et al., 2017, Nguyen et al., 2018]. Sometimes they adapt the loss function to remove or reduce the weight of the KL-divergence term, which reduces the pressure on weight variances to grow [Fortunato et al., 2018]. Other times researchers pretrain the means with just the NLL loss, which makes it possible to stop training after relatively little training on the ELBO loss, which stops the variances from growing too much [Nguyen et al., 2018]. Another approach, which we have not seen tried, would be to use a very tight prior, effectively enforcing the desire to have a basically deterministic network (a prior inversely proportional to $\sqrt{D}$ would balance the soap-bubble variance). However, this sort of very tight prior is not compatible with the use of data-dependent priors in sequential learning.

For most of these tweaks, the resulting network is not fully optimizing the ELBO. This does not necessarily make the resulting network useless—after all the ELBO is only a bound on the actual model evidence, and other methods like Deep Ensembles work surprisingly well despite not necessarily estimating the model posterior at all. However, if we have a theoretically principled way to fix our sampling problems without resorting to ad-hoc tweaks, then we should prefer that. Radial BNNs offer exactly that theoretically principled fix.

6 Discussion

Bayesian neural networks need to scale to large models in order to reach their full potential. Until now, researchers who want BNNs at scale needed to accept a posterior distribution which has zero probability mass almost everywhere—an implausible and impractical assumption. At the same time, MFVI requires increasingly demanding ad-hoc tweaks in order to work in anything but small models. We show why MFVI faces a serious gradient estimation problem which gets

worse in high dimensions. Based on this motivation, we introduce Radial BNNs. This alternative variational inference posterior approximation is simple to implement, computationally fast, robust to hyperparameters, and scales to large models. Radial BNNs outperform other efficient BNN methods, and have the potential to craft data-dependent priors for use in applications like continual learning.

7 Acknowledgements

We would like to especially acknowledge Alexander Lvovsky for spotting an error in a proof and Lewis Smith for useful conversations and suggestion of the experiment in Appendix F. We would also like to thank Milad Alizadeh, Joost van Amersfoort, Gregory Farquhar, Angelos Filos, and Andreas Kirsch for valuable discussions and/or comments on drafts. In addition, we gratefully thank the Alan Turing Institute and Google for their donation of computing resources. Lastly, we thank the EPSRC for their support of Sebastian Farquhar through the Centre for Doctoral Training in Cyber Security at the University of Oxford.

References

Chris Bishop. Introduction. In *Pattern Recognition and Machine Learning*. Springer, 2006. 1, 3, A

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight Uncertainty in Neural Networks. *Proceedings of the 32nd International Conference on Machine Learning*, 37:1613–1622, 2015. 1, 2, 3.1, 3.2, 5.1, D.1

Arslan Chaudhry, Puneet K. Dokania, Thalaiyasingam Ajanthan, and Philip H. S. Torr. Riemannian Walk for Incremental Learning: Understanding Forgetting and Intransigence. *ECCV*, 2018. 4.2, D.2, D.3

Sebastian Farquhar and Yarin Gal. Towards Robust Evaluations of Continual Learning. *Lifelong Learning: A Reinforcement Learning Approach Workshop ICML*, May 2018a. arXiv: 1805.09733. 4.2, D.2, D.3

Sebastian Farquhar and Yarin Gal. A Unifying Bayesian View of Continual Learning. *Bayesian Deep Learning Workshop at NeurIPS*, 2018b. 4.2

Angelos Filos, Sebastian Farquhar, Aidan N Gomez, Tim G J Rudner, Zachary Kenton, Lewis Smith, Milad Alizadeh, Arnoud de Kroon, and Yarin Gal. Benchmarking Bayesian Deep Learning with Diabetic Retinopathy Diagnosis. 2019. 4.1

Meire Fortunato, Charles Blundell, and Oriol Vinyals. Bayesian Recurrent Neural Networks. *arXiv preprint*, 2017. 5.1

Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Ian Osband, Alex Graves, Vlad Mnih, Remi Munos, Demis Hassabis, Olivier Pietquin, Charles Blundell, and Shane Legg. Noisy Networks for Exploration. *ICLR*, 2018. 6, 5.1

Yarin Gal. Uncertainty in Deep Learning. *PhD Thesis*, 2016. B

Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. *Proceedings of the 33rd International Conference on Machine Learning*, 48: 1050–1059, 2015. 1, 2, 4.2, 2

Alex Graves. Practical Variational Inference for Neural Networks. *Neural Information Processing Systems*, 2011. 1, 2, 3.1

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *CVPR*, 7(3):171–180, 2016. D.2

José Miguel Hernández-Lobato and Ryan P. Adams. Probabilistic Backpropagation for Scalable Learning of Bayesian Neural Networks. *Proceedings of the 32nd International Conference on Machine Learning*, 2015. 4

Geoffrey Hinton and Drew van Camp. Keeping Neural Networks Simple by Minimizing the Description Length of the Weights. *Proceedings of the 6th Annual ACM Conference on Computational Learning Theory*, 1993. 1, 2

Kaggle. Diabetic Retinopathy Detection, 2015. URL `https://www.kaggle.com/c/diabetic-retinopathy-detection/data`. 4.1

Mohammad Emtiyaz Khan, Didrik Nielsen, Voot Tangkaratt, Wu Lin, Yarin Gal, and Akash Srivastava. Fast and Scalable Bayesian Deep Learning by Weight-Perturbation in Adam. *International Conference on Machine Learning*, 2018. 2

Diederik P. Kingma, Maz Welling, and Soumith Chintala. Auto-Encoding Variational Bayes. *International Conference on Learning Representations*, 2014. 3.1, B

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, March 2017. 1, 4, 4.2

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. 2016. 1, 2, 4.2

Christian Leibig, Vaneeda Allken, Murat Seçkin Ayhan, Philipp Berens, and Siegfried Wahl. Leveraging un-

certainty information from deep neural networks for disease detection. *Scientific Reports*, 7(1), December 2017. 1, 4.1, 4.1.1, D.1

Christos Louizos and Max Welling. Structured and Efficient Variational Deep Learning with Matrix Gaussian Posteriors. *International Conference on Machine Learning*, pages 1708–1716, 2016. 1, 2, 3.3, 2

David J. C. MacKay. A Practical Bayesian Framework for Backpropagation Networks. *Neural Computation*, 4(3):448–472, 1992. 2

George Marsaglia. Choosing a Point from the Surface of a Sphere. *The Annals of Mathematical Statistics*, 43 (2):645–646, April 1972. ISSN 0003-4851, 2168-8990. doi: 10.1214/aoms/1177692644. 3.1

Angel Muleshkov and Tan Nguyen. Easy Proof of the Jacobian for the N-Dimensional Polar Coordinates. *Pi Mu Epsilon Journal*, 14:269–273, 2016. B

Mervin E. Muller. A Note on a Method for Generating Points Uniformly on N-dimensional Spheres. *Commun. ACM*, 2(4):19–20, April 1959. ISSN 0001-0782. doi: 10.1145/377939.377946. 3.1

Radford M. Neal. *Bayesian learning for neural networks*. PhD Thesis, University of Toronto, 1995. 2

Cuong V. Nguyen, Yingzhen Li, Thang D. Bui, and Richard E. Turner. Variational Continual Learning. *International Conference on Learning Representations*, 2018. 1, 4, 4.1.2, 4.2, 5.1, D.2

ChangYong Oh, Efstratios Gavves, and Max Welling. BOCK : Bayesian Optimization with Cylindrical Kernels. *Proceedings of The 35th International Conference on Machine Learning*, pages 3868–3877, June 2018. arXiv: 1806.01619. 7

Changyong Oh, Kamil Adamczewski, and Mijung Park. Radial and Directional Posteriors for Bayesian Neural Networks. *arXiv*, February 2019. arXiv: 1902.02603. 1, 2

Kazuki Osawa, Siddharth Swaroop, Anirudh Jain, Runa Eschenhagen, Richard E. Turner, Rio Yokota, and Mohammad Emtiyaz Khan. Practical Deep Learning with Bayesian Principles. *arXiv:1906.02506 [cs, stat]*, June 2019. arXiv: 1906.02506. 2

Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. On the Convergence of Adam and Beyond. *ICLR*, February 2018. D.2

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. *Proceedings of The 31st International Conference on Machine Learning*, 2014. 3.1, B

Hippolyt Ritter, Aleksandar Botev, and David Barber. A Scalable Laplace Approximation for Neural networks. page 15, 2018. 4.2

Shengyang Sun, Changyou Chen, and Lawrence Carin. Learning Structured Weight Uncertainty in Bayesian Neural Networks. *Artificial Intelligence and Statistics*, pages 1283–1292, 2017. 1, 2, 3.3, 2

Shengyang Sun, Guodong Zhang, Jiaxin Shi, and Roger Grosse. Functional variational Bayesian Neural Networks. *International Conference on Learning Representations*, 2019. 1, 2, 2

Max Welling and Yee Whye Teh. Bayesian Learning via Stochastic Gradient Langevin Dynamics. *Proceedings of the 28th International Conference on Machine Learning*, 2011. 1, 2

Anqi Wu, Sebastian Nowozin, Edward Meeds, Richard E Turner, Jose Miguel Hernandez-Lobato, and Alexander L Gaunt. Deterministic Variational Inference for Robust Bayesian Neural Networks. *International Conference on Learning Representations*, 2019. 1, 2, 2

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv:1708.07747 [cs, stat]*, August 2017. 4, 4.2

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual Learning Through Synaptic Intelligence. *Proceedings of the 34th International Conference on Machine Learning*, pages 3987–3995, 2017. 4.2

Appendices

A Understanding the Soap Bubble

The emergence of a 'soap-bubble' is a well-known property in multi-variate Gaussian distributions as the number of dimensions increases (e.g., see Bishop [2006]). The observation is that, even though the highest probability density is near the mean, because there is just *so much more volume* further from the mean in high-dimensional spaces it ends up being the case that most of the probability mass is far from the mean.

One way to understand this is to examine the probability density function of the multivariate Gaussian along its radius. Consider a D-dimensional isotropic Gaussian. We examine a thin shell with thickness η, which tends to zero, at distance r between a sampled point, $\mathbf{w}$, and the mean of the multivariate Gaussian, $\boldsymbol{\mu}$. The probability density function over the radius is given by:

$$\lim_{\eta \to 0} p(r - \eta < \|\mathbf{w} - \boldsymbol{\mu}\| < r + \eta) = \frac{S_D}{(2\pi\sigma^2)^{D/2}} \cdot r^{D-1} \cdot e^{-\frac{r^2}{2\sigma^2}} \tag{6}$$

where S_D is the surface area of a hypersphere in a D-dimensional space.

The first term of the product is just a normalizing constant (S_D is the surface area of a D-dimensional hypersphere).

The second term, r^{D-1}, reflects the growing *volume* in shells away from the origin. In the region where r is small, and for the large D found in BNNs with many parameters, this term (red in figure 8) dominates and drives the probability density towards zero.

The exponential term $e^{-\frac{r^2}{2\sigma^2}}$ reflects the Gaussian density (inverse shown in green in figure 8). For larger r the exponential term becomes very small and drives the probability density towards zero. Almost all the probability mass is in the 'soap-bubble' in the region where neither term becomes very small. We consider the isotropic case here for simplicity, but the non-isotropic Gaussian has a similar soap-bubble.[7]

B Derivation of the Entropy Term of the KL-divergence

In this section, we show that the component of KL-divergence term of the loss which is the entropy of the

[7] Oh et al. [2018] consider soap-bubbles in Bayesian optimization. But it has not been considered for MFVI.

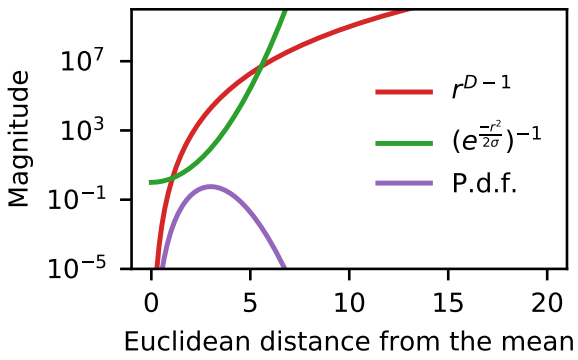

Figure 8: We can understand the soap-bubble by looking at components of the p.d.f. in eq. 6. For small r the volume term (red) dominates and the normalized p.d.f. is very small. For big r the Gaussian density term (inverse shown in green) dominates and the p.d.f. is small again. Almost all the probability mass is in the intermediate region where neither term dominates: the 'soap bubble'. The intermediate region becomes narrower and further from the mean as D is bigger. Here we show $D = 10$.

posterior distribution over the weights $q(\mathbf{w}^{(x)})$ can be estimated as:

$$\mathcal{L}_{\text{entropy}} := \int q(\mathbf{w}^{(x)}) \log[q(\mathbf{w}^{(x)})] d\mathbf{w}^{(x)} \tag{7}$$

$$= -\sum_i \log[\sigma_i^{(x)}] + \text{const} \tag{8}$$

where i is an index over the weights of the model.

Throughout this section we use a superscript indicates the basis—an (x) means we are in the Cartesian coordinate system tied to the weight-space while (r) is the hyperspherical coordinate system (the letter is the canonical 'first' coordinate of that coordinate system).

We begin by applying the reparameterization trick [Kingma et al., 2014, Rezende et al., 2014]. Following the auxiliary variable formulation of Gal [2016], we express the probability density function of $q(\mathbf{w}^{(x)})$ with an auxiliary variable.

$$q(\mathbf{w}^{(x)}) = \int q(\mathbf{w}^{(x)}, \boldsymbol{\epsilon}^{(r)}) d\boldsymbol{\epsilon}^{(r)} \tag{9}$$

$$= \int q(\mathbf{w}^{(x)} | \boldsymbol{\epsilon}^{(r)}) q(\boldsymbol{\epsilon}^{(r)}) d\boldsymbol{\epsilon}^{(r)} \tag{10}$$

$$= \int \delta(\mathbf{w}^{(x)} - g(\mu, \sigma, \boldsymbol{\epsilon}^{(r)})) q(\boldsymbol{\epsilon}^{(r)}) d\boldsymbol{\epsilon}^{(r)}. \tag{11}$$

In equation (11), we have used a reparameterization

trick transformation:

$$g(\mu, \sigma, \boldsymbol{\epsilon}^{(r)}) = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \mathbf{T}_{rx}(\boldsymbol{\epsilon}^{(r)}) \tag{12}$$

where $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ are parameters of the model and where $\mathbf{T}_{rx}$ is the standard transformation from hyperspherical into Cartesian coordinates.

Substituting equation (11) into the definition of the entropy loss term in equation (7), and applying the definition of the Kronecker delta we can eliminate dependence on $\mathbf{w}^{(x)}$:

$$\mathcal{L}_{\text{entropy}} = \int q(\mathbf{w}^{(x)}) \log[q(\mathbf{w}^{(x)})] d\mathbf{w}^{(x)} \tag{13}$$

$$= \int \left(\int \delta(\mathbf{w}^{(x)} - g(\boldsymbol{\mu}, \boldsymbol{\sigma}, \boldsymbol{\epsilon}^{(r)}) \right. \\ \left. q(\boldsymbol{\epsilon}^{(r)}) d\boldsymbol{\epsilon}^{(r)} \right) \log[q(\mathbf{w}^{(x)})] d\mathbf{w}^{(x)} \tag{14}$$

$$= \int q(\boldsymbol{\epsilon}^{(r)}) \log[q(g(\boldsymbol{\mu}, \boldsymbol{\sigma}, \boldsymbol{\epsilon}^{(r)}))] d\boldsymbol{\epsilon}^{(r)}. \tag{15}$$

Then, we perform a coordinate transformation from $g(\boldsymbol{\mu}, \boldsymbol{\sigma}, \boldsymbol{\epsilon}^{(r)})$ to $\boldsymbol{\epsilon}^{(r)}$ using the Jacobian of the transformation and simplify.

$$= \int q(\boldsymbol{\epsilon}^{(r)}) \log \left[q(\boldsymbol{\epsilon}^{(r)}) \left| \frac{\partial g(\boldsymbol{\mu}, \boldsymbol{\sigma}, \boldsymbol{\epsilon}^{(r)})}{\partial \boldsymbol{\epsilon}^{(r)}} \right|^{-1} \right] d\boldsymbol{\epsilon}^{(r)} \tag{16}$$

$$= \int q(\boldsymbol{\epsilon}^{(r)}) \log \left[q(\boldsymbol{\epsilon}^{(r)}) \left| \prod_i \sigma_i^{(x)} \frac{\partial \epsilon_i^{(x)}}{\partial \epsilon_j^{(r)}} \right|^{-1} \right] d\boldsymbol{\epsilon}^{(r)} \tag{17}$$

$$= \int q(\boldsymbol{\epsilon}^{(r)}) \log \left[q(\boldsymbol{\epsilon}^{(r)}) \left| \text{diag}(\boldsymbol{\sigma}) \frac{\partial \epsilon_i^{(x)}}{\partial \epsilon_j^{(r)}} \right|^{-1} \right] d\boldsymbol{\epsilon}^{(r)} \tag{18}$$

$$= \int q(\boldsymbol{\epsilon}^{(r)}) \log \left[\frac{q(\boldsymbol{\epsilon}^{(r)})}{\prod_i \sigma_i^{(x)}} \left| \frac{\partial \epsilon_i^{(x)}}{\partial \epsilon_j^{(r)}} \right|^{-1} \right] d\boldsymbol{\epsilon}^{(r)} \tag{19}$$

In the last line we have used the fact that $\forall i : \sigma_i^{(x)} \geq 0$ allowing us to pull the determinant of this diagonal matrix out.

$\left| \frac{\partial \epsilon_i^{(x)}}{\partial \epsilon_j^{(r)}} \right|$ is the determinant of the Jacobian for the transformation from Cartesian to hyperspherical coordinates for which we use the result by Muleshkov and Nguyen [2016]:

$$\left| \frac{\partial \boldsymbol{\epsilon}_i^{(x)}}{\partial \boldsymbol{\epsilon}_j^{(r)}} \right| = \text{abs} \left(\big(-1 \big)^{D-1} \big(\epsilon_0^{(r)} \big)^{D-1} \prod_{i=2}^{D} \big(\sin(\epsilon_i^{(r)}) \big)^{i-1} \right). \tag{20}$$

We know that $\epsilon_0^{(r)} \geq 0$ because the radial dimension in hyperspherical coordinates can be assumed positive without loss of generality. We also know $0 \leq \epsilon_i^{(r)} \leq \pi$ for $2 \leq i \leq D$ for the hyperspherical coordinate system. So we can simplify the signs:

$$= \big(\epsilon_0^{(r)} \big)^{D-1} \prod_{i=2}^{D} \big(\sin(\epsilon_i^{(r)}) \big)^{i-1}. \tag{21}$$

Therefore, plugging equation (21) into (19):

$$\mathcal{L}_{\text{entropy}} = \int q(\boldsymbol{\epsilon}^{(r)}) \log \left[\frac{q(\boldsymbol{\epsilon}^{(r)})}{\text{abs}(\prod_i \sigma_i^{(x)})} \left| \frac{\partial \epsilon_i^{(x)}}{\partial \epsilon_j^{(r)}} \right|^{-1} \right] d\boldsymbol{\epsilon}^{(r)} \tag{22}$$

$$= \int q(\boldsymbol{\epsilon}^{(r)}) \log[q(\boldsymbol{\epsilon}^{(r)})] \\ - \log[\text{abs}(\prod_i \sigma_i^{(x)})] \\ - \log \left[\big(\epsilon_0^{(r)} \big)^{D-1} \prod_{i=2}^{D} \big(\sin(\epsilon_i^{(r)}) \big)^{i-1} \right] d\boldsymbol{\epsilon}^{(r)}. \tag{23}$$

Very simply, we can observe that only the middle term depends on the parameters and we must therefore only compute this term in order to compute gradients. For sake of completeness, we address the other integrals below, in case one wants to have the full value of the loss (though since it is a lower bound in any case, the full value is not very useful).

The probability density function of the noise variable is separable into independent distributions. The distribution of $\epsilon_0^{(r)}$ is a unit Gaussian. The angular dimensions are distributed so that sampling is uniform over the hypersphere. However, this does **not** mean that the distribution over each angle is uniform, as this would lead to bunching near the n-dimensional generalization of the poles. (Intuitively, there is more surface area per unit of angle near the equator, as is familiar from cartography.) Instead, we use the fact that the area element over the hypersphere is:

$$dA = d\epsilon_D^{(r)} \prod_{i=1}^{D-1} \sin(\epsilon_i^{(r)})^{D-i} d\epsilon_i^{(r)} \tag{24}$$

where we remember that $\epsilon_D^{(r)}$ is between $-\pi$ and π, and the rest of the angular elements of $\boldsymbol{\epsilon}^{(r)}$ are between 0 and π. The resulting probability density function is:

$$q(\boldsymbol{\epsilon}^{(r)}) = \prod_{i=0}^{D} q(\epsilon_i^{(r)}) \tag{25}$$

$$= \frac{1}{\sqrt{2\pi}} e^{-\frac{\epsilon_0^2}{2}} \cdot \prod_{i=1}^{D-1} \sin(\epsilon_i^{(r)})^{D-i}. \tag{26}$$

As a result, all three of the terms in equation (23) are analytically tractable. Inserting the probability density function from equation (26) into the first term of the loss, splitting up the product inside the logarithm, and separating independent terms we get:

$$\begin{aligned}
&\int q(\boldsymbol{\epsilon}^{(r)}) \log[q(\boldsymbol{\epsilon}^{(r)})] d\boldsymbol{\epsilon}^{(r)} \\
&= \int_0^\infty d\epsilon_0^{(r)} \int_{-\pi}^{\pi} d\epsilon_D^{(r)} \int_0^{\pi} \prod_{i=1}^{D-1} d\epsilon_i^{(r)} \\
&\quad \cdot \frac{1}{\sqrt{2\pi}} e^{-\frac{\epsilon_0^2}{2}} \cdot \prod_{i=1}^{D-1} \sin(\epsilon_i^{(r)})^{D-i} \\
&\quad \cdot \log\Big[\frac{1}{\sqrt{2\pi}} e^{-\frac{\epsilon_0^2}{2}} \cdot \prod_{i=1}^{D-1} \sin(\epsilon_i^{(r)})^{D-i}\Big] \qquad (27) \\
&= \int_0^\infty d\epsilon_0^{(r)} \int_{-\pi}^{\pi} d\epsilon_D^{(r)} \int_0^{\pi} \prod_{i=1}^{D-1} d\epsilon_i^{(r)} \\
&\quad \cdot \frac{1}{\sqrt{2\pi}} e^{-\frac{\epsilon_0^2}{2}} \cdot \prod_{i=1}^{D-1} \sin(\epsilon_i^{(r)})^{D-i} \\
&\quad \cdot \log\Big[\frac{1}{\sqrt{2\pi}} e^{-\frac{\epsilon_0^2}{2}}\Big] + \sum_{i=1}^{D-1} \log\big[\sin(\epsilon_i^{(r)})^{D-i}\big] \qquad (28) \\
&= \int_0^\infty d\epsilon_0^{(r)} \frac{1}{\sqrt{2\pi}} e^{-\frac{\epsilon_0^2}{2}} \log\Big[\frac{1}{\sqrt{2\pi}} e^{-\frac{\epsilon_0^2}{2}}\Big] \\
&\quad + \int_{-\pi}^{\pi} d\epsilon_D^{(r)} \\
&\quad + \sum_{i=1}^{D-1} \int_0^{\pi} d\epsilon_i^{(r)} \sin(\epsilon_i^{(r)})^{D-i} \log\big[\sin(\epsilon_i^{(r)})^{D-i}\big] \qquad (29) \\
&= -\frac{\log[2\pi]+1}{4} + 2\pi \\
&\quad + \sum_{i=1}^{D-1} \int_0^{\pi} d\epsilon_i^{(r)} \sin(\epsilon_i^{(r)})^{D-i} \log\big[\sin(\epsilon_i^{(r)})^{D-i}\big] \qquad (30)
\end{aligned}$$

We can simplify the first two of the terms in equation 29, but the third is difficult to solve in general (though tractable for any specific D). Regardless, this term is constant and therefore not needed for our optimization.

Inserting the probability density function from equation (26) into the second term of the loss in equation (23) we get

$$\begin{aligned}
\int q(\boldsymbol{\epsilon}^{(r)}) \log[\prod_i \sigma_i^{(x)}] d\boldsymbol{\epsilon}^{(r)} &= \log[\prod_i \sigma_i^{(x)}] \qquad (31) \\
&= \sum_i \log[\sigma_i^{(x)}]. \qquad (32)
\end{aligned}$$

This second term is identical to the entropy of the multivariate Gaussian variational posterior typically used in MFVI.

And for the third term we begin by expanding the logarithm and simplifying:

$$\begin{aligned}
&\int q(\boldsymbol{\epsilon}^{(r)}) \log\Big[\big(\epsilon_0^{(r)}\big)^{D-1} \prod_{i=2}^{D} \big(\sin(\epsilon_i^{(r)})\big)^{i-1}\Big] d\boldsymbol{\epsilon}^{(r)} \qquad (33) \\
&= (D-1) \int_0^\infty q(\epsilon_0^{(r)}) \log[\epsilon_0^{(r)}] d\epsilon_0^{(r)} \\
&\quad + \sum_{i=2}^{D} \frac{(i-1)}{\pi} \int_0^{\pi} \log\Big[\sin(\epsilon_i^{(r)})\Big] d\epsilon_i^{(0)} \qquad (34) \\
&\qquad (35)
\end{aligned}$$

and then inserting the p.d.f. from equation (26) and solving analytically tractable integrals:

$$\begin{aligned}
&= \frac{(D-1)}{\sqrt{2\pi}} \int_0^\infty e^{-\frac{\left(\epsilon_0^{(r)}\right)^2}{2}} \log[\epsilon_0^{(r)}] d\epsilon_0^{(r)} \\
&\quad + \sum_{i=2}^{D} \frac{(i-1)}{\pi} \cdot -\pi \log[2] \qquad (36) \\
&= \frac{(D-1)}{\sqrt{2\pi}} \cdot -\frac{1}{2}\sqrt{\frac{\pi}{2}}(\gamma + \log[2]) - \sum_{i=2}^{D}(i-1) \cdot \log[2] \qquad (37) \\
&= -\frac{(D-1)}{4} \cdot (\gamma + \log[2]) - \frac{(D-1)(D-2)}{2} \log[2] \qquad (38) \\
&= -\frac{(D-1)\gamma}{4} - \frac{(D-1)(2D-3)}{4} \log[2]. \qquad (39) \\
&\qquad (40)
\end{aligned}$$

where γ is the Euler-Mascheroni constant. This is, again, constant and may be neglected for optimization.

As a result, we can minimize the entropy term of the loss simply by finding

$$\mathcal{L}_{\text{entropy}} = -\sum_i \log[\sigma_i^{(x)}] + \text{const} \qquad (41)$$

C Setting a Radial Prior

In most of our experiments, we use a typical multivariate Gaussian unit prior in order to ensure comparability with prior work. However, in some settings, such as the Variational Continual Learning setting, it is useful to use the radial posterior as a prior. We begin similarly to the previous derivation, with all unchanged expect that we are estimating

$$\mathcal{L}_{\text{cross-entropy}} = \int q(\mathbf{w}^{(x)}) \log[p(\mathbf{w}^{(x)})] d\mathbf{w}^{(x)}. \qquad (42)$$

The derivation proceeds similarly until equation (23), and the second and third terms are identical except the second term taking a product over elements of $\boldsymbol{\sigma}^{(x)}_{(\text{prior})}$ of the prior, not the posterior.

Evaluating the gradient of the log probability density function of the prior depends only on the radial term, since the distribution is uniform in all angular dimensions. We therefore find

$$\int q(\boldsymbol{\epsilon}^{(r)}) \log[p\Big(\frac{\mathbf{w}^{(x)} - \boldsymbol{\mu}^{(x)}_{(\text{prior})}}{\boldsymbol{\sigma}^{(x)}_{(\text{prior})}}\Big)] d\boldsymbol{\epsilon}^{(r)}. \tag{43}$$

Rather than solve the integral, we can estimate this as a Monte Carlo approximation:

$$\approx \frac{1}{N}\sum_{i=1}^{N} -\frac{1}{2}\left\|\frac{\mathbf{w}^{(x)} - \boldsymbol{\mu}^{(x)}_{(\text{prior})}}{\boldsymbol{\sigma}^{(x)}_{(\text{prior})}}\right\|^2. \tag{44}$$

By adding the three terms we estimate the cross-entropy term of the ELBO loss function.

D Experimental Settings

D.1 Diabetic Retinopathy Settings

The diabetic retinopathy data are publicly available at `https://www.kaggle.com/c/diabetic-retinopathy-detection/data`. We augment and preprocess them similarly to Leibig et al. [2017]. The images for our main experiments in §4.1 are downsampled to 512x512 while the smaller robustness experiment in §4.1.3 uses images downsampled to 256x256 We randomly flip horizontally and vertically. Then randomly rotate 180 degrees in either direction. Then we pad by between 0 and 5% of the width and height and randomly crop back down to the intended size. We then randomly crop to between 90% and 110% of the image size, padding with zeros if needed. We finally resize again to the intended size and normalize the means and standard deviations of each channel separately based on the training set means and standard deviations. The training set has 44,594 RGB images. There are 7,026 validation and 10,000 test images.

The smaller model used for robustness experiments is loosely inspired by VGG-16, with only 16 channels, except that it is a Bayesian neural network with mean and standard deviations for each weight, and that instead of fully connected networks at the end it uses a concatenated global mean and average pool. The larger model used in the main experiments is VGG-16 but with the concatenated global mean and average pool instead of fully connected layers as above. The only difference is that we use only 46 channels, rather than 64 channels as in VGG-16, because the BNN has twice as many parameters as a similarly sized deterministic network and we wanted to compare models with the same number of parameters. For the dropout model we use VGG-16 with the full 64 channels, and similarly for each of the models in the deep ensemble. The prior for training MFVI and Radial BNNs was a unit multivariate Gaussian. (We also tried using the scale mixture prior used in Blundell et al. [2015] and found it made no difference.) Instead of optimizing σ directly we in fact optimize ρ such that $\sigma = \log(1 + e^{\rho})$ which guarantees that σ is always positive. In some cases, as described in the paper, the first epoch only trained the means and uses a NLL loss function. This helps the optimization, but in principle can still allow the variances to train fully if early stopping is not employed (unlike reweighting the KL-divergence). Thereafter we trained using the full ELBO loss over all parameters. Unlike some prior work using MFVI, we have not downweighted the KL-divergence during training.

For the larger models, we searched for hyperparameters using Bayesian optimization. We searched between 0 and -10 as the initial value of ρ (equivalent to σ values of $\log(2)$ and $2 \cdot 10^{-9}$). For the learning rate we considered 10^{-3} to 10^{-5} using Adam with a batch size of 16. Otherwise, hyperparameters we based on exploration from the smaller model.

We then computed the test scores using a Monte Carlo estimate from averaging 16 samples from the variational distribution. We estimate the model's uncertainty about a datapoint using the mutual information between the posterior's parameters and the prediction on a datapoint. This estimate is used to rank the datapoints in order of confidence and compute the model's accuracy under the assumption of referring increasingly many points to medical experts.

For the smaller models, we performed an extensive random hyperparameter search. We tested each configuration with both MFVI and Radial BNNs. We tested each configuration for both an SGD optimizer and Amsgrad. When training with SGD we used Nesterov momentum 0.9 and uniformly sampled from 0.01, 0.001 and 0.0001 as learning rates, with a learning rate decay each epoch of either 1.0 (no decay), 0.98 or 0.96. When training with Amsgrad we uniformly sampled from learning rates of 0.001, 0.0001, and 0.00001 and did not use decay. We uniformly selected batch sizes from 16, 32, 64, 128, and 256. We uniformly selected the number of variational distribution samples used to estimate the loss from 1, 2, and 4. However, because we discarded all runs where there was insufficient graphics memory, we were only able to test up to 64x4 or 256x1 and batch sizes above 64 were proportionately less likely to appear in the final results. We selected the

initial variance from ρ values of -6, -4, -2, or 0. We also tried reducing the number of convolutional channels by a factor of 5/8 or 3/8 and found that this did not seem to improve performance. We ran our hyperparameter search runs for 150 epochs. We selected the best hyperparameter configurations based on the best validation accuracy at any point during the training. We trained the models for 500 epochs but selected the models saved from 300 epochs as all models had started to overfit by the end of training. For MFVI, this was using the SGD optimizer with learning rate 0.001, decay rate 0.98 every epoch, batch size 16, 4 variational samples for estimating the loss during training and ρ of -6. This outperformed the others by a significant margin. Using our code on a V100 GPU with 8 vCPUs and an SSD this took slightly over 13 hours to train each model. For the radial posterior, this was the Adam optimizer with learning rate 0.0001, batch size 64, 1 variational sample for estimating the loss during training and a ρ of -6. Using our code on the same GPU, this took slightly over 3h to run. However, for the radial posterior there were very many other configurations with similar validation accuracies (one of the advantages of the posterior).

For the experiment shown in Figure 7, we have selected slightly different hyperparameters in order to train more quickly. For both models, we use Adam with learning rate 0.0001 and train for 500 epochs. The models have 5/8 the number of channels of VGG-16. The models are trained with batch size 64 and 4 variational samples to estimate the loss and its standard deviation.

D.2 Variational Continual Learning Settings

We build on the code provided by Nguyen et al. [2018] at `https://github.com/nvcuong/variational-continual-learning` adapted for FashionMNIST. The FashionMNIST dataset was downloaded using pytorch's built in vision datasets. The data were normalized by subtracting the training set mean and dividing by the training set standard deviation.

The classes are ordered in the conventional order. The model is initialized randomly—without pretraining the means (unlike Nguyen et al. [2018]). The model is then trained on the first two classes. The weights are carried over to the next task and set as a prior, while the model is trained on the next two classes, and so on. Note that we perform the tasks in a multi-headed way—each task has its own output head. This may not be an ideal exemplar of the continual learning problem [Chaudhry et al., 2018, Farquhar and Gal, 2018a] but it forms an effective test of the posterior. We do *not* use coresets, unlike Nguyen et al. [2018], as this would *not* form an effective test of the quality of the posterior.

Models are Bayesian MLPs with four hidden layers with 200 units in each. The prior for training was a unit multivariate Gaussian. Instead of optimizing σ directly we in fact optimize ρ such that $\sigma = \log(1 + e^{\rho})$ which guarantees that σ is always positive. Models are optimized using Amsgrad [Reddi et al., 2018] with learning rate 0.001 with shuffling and discarding final incomplete batches each epoch. We perform a grid search over the number of epochs each task is trained over (3, 5, 10, 15, 20, 60, 120) and batch sizes (1024, 2048, 10000). We used 90% of the standard training dataset (54000 points) as a training dataset, with 10% (6000 points) withheld as a validation dataset. We initialize ρ to -6 and use the initialization by He et al. [2016] for the means. The radial posterior would work with a much larger ρ, but we wanted to use comparable initializations for each. We optimized for average validation accuracy over all models on the final task. We used the standard 10000 points as a test dataset. The best configuration for the MFVI posterior was found to be 60 epochs of batch size 1024 (note that this differs from the 120 epochs of batch size 12000 reported in Nguyen et al. [2018] perhaps because they pretrain the means). The best configuration for the radial posterior was found to be 20 epochs of batch size 1024. We report the individual accuracies for each head on the test dataset.

D.3 Single-headed FashionMNIST continual learning

Previous authors have noted that for *continual learning* the single-headed environment—where the model has a single output head shared over all tasks and must therefore identify the task as well as the output based on the input—is a much harder task, possibly more reflective of continual learning [Chaudhry et al., 2018, Farquhar and Gal, 2018a]. While the multi-headed setting suffices to demonstrate improvement to the posterior, we offer some results for the single-headed setting here in the appendix for the interest of continual learning specialists, though we do not find that our posterior solves the problem.

We perform a similar grid search as before, selecting the hyperparameters that offer the highest average validation set accuracy for the final model over all five tasks. Note that in our grid search each task gets the same hyperparameters, reflecting the idea that the task distribution is not known in advance.

Our Radial BNN does not solve the continual learning single-headed problem, but it does show improved performance relative to the MFVI baseline. As we show in Figure 9, the Radial BNN shows some remembering

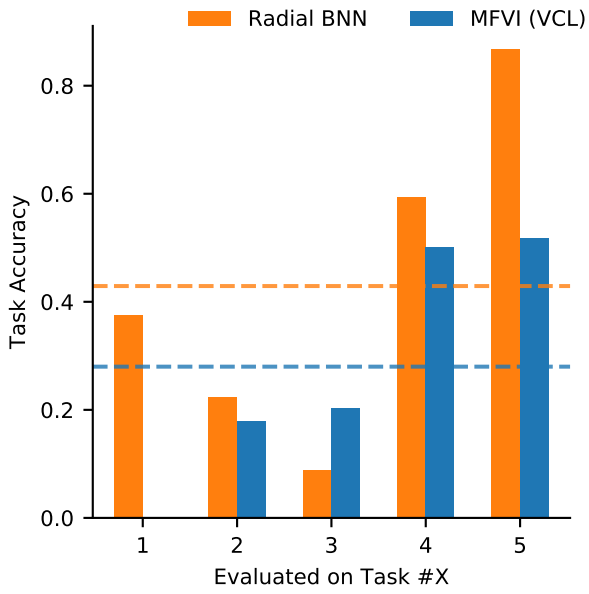

Figure 9: FashionMNIST. In the single-headed setting, where all training and testing ignores the task label, the situation is more difficult. MFVI (VCL) forgets tasks—any hyperparameter configuration that allows it to fully learn the most recent task makes it forget old tasks completely. The shown run offers the best average accuracy over all tasks for the last model. Our Radial BNN preserves information somewhat better even while training to a higher accuracy on the final task. Average accuracy is the dotted line.

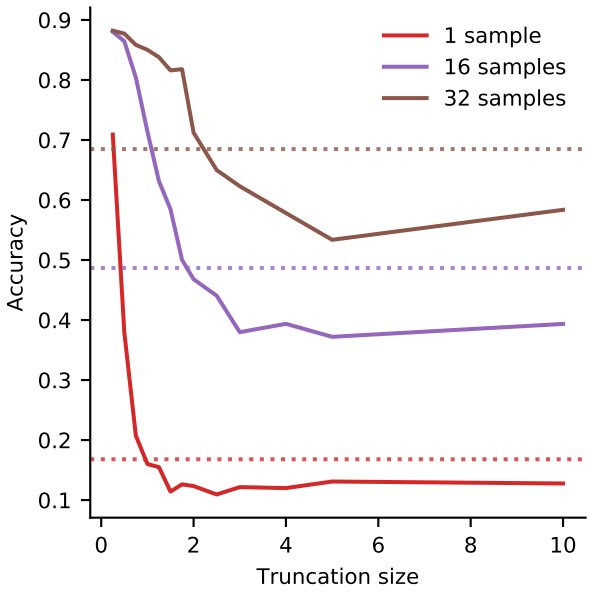

Figure 10: Dotted lines show untruncated Gaussian performance. Highly truncated Gaussians *improve* MFVI. This effect is most significant when small numbers of samples from the posterior are used to estimate the gradient. We conclude that despite bias, the low variance offered by truncation improves gradient estimates. Results averaged over 10 initial seeds for each truncation size.

on old tasks (which includes identifying the task that the image comes from). Moreover it is able to maintain good accuracy on the newest task. Meanwhile, the hyperparameters that allow MFVI to optimize last-task average accuracy mean it learns a very uncertain model which has bad accuracy on the newest tasks. This is because hyperparameters that would let it learn a high-accuracy model for the newest task would cause it to forget everything it saw earlier.

E Results on the UCI datasets

We do not believe that the standard UCI Bayesian learning experiments which are heavily used in the field offer much insight in this case. This is because all of the problems have low dimension (4-16) and because the experimental design allows only for a single hidden layer with 50 units. This is required because many of the expensive techniques that researchers develop and evaluate on the UCI datasets only scale to very small models and inputs.

For sake of completeness we show some results of our methods on the UCI datasets. As expected, our method does not outperform the more expensive techniques with complex covariances within the approximate posterior. Moreover, as expected our method performs very similarly to MFVI. In the small number of parameters involved in the UCI dataset experimental settings, the sampling problems for MFVI do not become severe. In this low-dimensional regime, we do not expect any particular advantage of Radial BNNs over MFVI, which is what we find.

Note that in some cases the MFVI and Radial BNN results we show are somewhat worse than those reported in other papers. We believe this is because of the fact that the resources devoted to hyperparameter search are not always the same in different papers. We only searched over learning rates of 0.001 and 0.0001 using Adam and batch sizes of 16, 64, and 1000. In the majority of datasets listed our results are competitive with what previous authors report for MFVI.

F Further Gradient Experiment

A further analysis demonstrates that we can improve the performance of MFVI models by using a low-variance but highly biased estimator of the NLL loss. We do this by estimating the NLL with a truncated version of the Gaussian sampling distribution, without changing how we analytically calculate the KL terms

Dataset	MFVI	Radial	Dropout	VMG	FBNN	PBP_MV	DVI
				Avg. Test LL and Std. Errors			
Boston	-2.58±0.06	-2.58± 0.05	-2.46±0.25	-2.46±0.09	-2.30±0.04	02.54±0.08	-2.41±0.02
Concrete	-5.08±0.01	-5.08±0.01	-3.04±0.09	-3.01±0.03	-3.10±0.01	-3.04±0.03	-3.06±0.01
Energy	-1.05±0.01	-0.91±0.03	-1.99±0.09	-1.06±0.03	**-0.68±0.02**	-1.01±0.01	-1.01 ± 0.06
Kin8nm	1.08±0.01	**1.35±0.00**	0.95±0.03	1.10±0.01	-	1.28±0.01	1.13±0.00
Naval	-1.57±0.01	-1.58±0.01	3.80±0.05	2.46±0.00	**7.13±0.02**	4.85±0.06	6.29±0.04
Pow. Plant	-7.54±0.00	-7.54±0.00	-2.80±0.05	-2.82±0.01	-	-2.78±0.01	-2.80±0.00
Protein	-3.67±0.00	-3.66±0.00	-2.89±0.01	-2.84±0.00	-2.89±0.00	-2.77±0.01	-2.85±0.01
Wine	-3.15±0.01	-3.15±0.01	-0.93±0.06	-0.95±0.01	-1.04±0.01	-0.97±0.01	**-0.90±0.01**
Yacht	-4.20±0.05	-4.20±0.05	-1.55±0.12	-1.30±0.02	-1.03±0.03	-1.64±0.02	**-0.47±0.03**
				Avg. Test RMSE and Std. Errors			
Boston	3.42±0.23	3.36±0.23	2.97±0.85	2.70±0.13	2.38±0.10	3.11±0.15	-
Concrete	5.71±0.15	5.62±0.14	5.23± 0.53	4.89±0.12	4.94±0.18	5.08±0.14	-
Energy	0.81±0.08	0.66±0.03	1.66±0.19	0.54±0.02	0.41±0.20	0.45±0.01	-
Kin8nm	0.37±0.00	0.16±0.00	0.10±0.00	0.08±0.00	-	**0.07±0.00**	-
Naval	0.01±0.00	0.01±0.00	0.01±0.00	0.00±0.00	0.00±0.00	0.00±0.00	-
Pow. Plant	4.02±0.04	4.04±0.04	4.02±0.18	4.04±0.04	-	3.91±0.14	-
Protein	4.40±0.02	4.34±0.03	4.36±0.04	4.13±0.02	4.33±0.03	3.94±0.02	-
Wine	0.65±0.01	0.64±0.01	0.62±0.04	0.63±0.01	0.67±0.01	0.64±0.01	-
Yacht	1.75±0.42	1.86±0.37	1.11±0.38	0.71±0.05	**0.61±0.07**	0.81±0.06	-

Table 2: Avg. test RMSE, predictive log-likelihood and s.e. for UCI regression datasets. Bold where one model is better than the next best ± their standard error. Results are from multiple papers and *hyperparameter search is not consistent*. MFVI and Radial are our implementations of standard MFVI and our proposed model respectively. Dropout is Gal and Ghahramani [2015]. Variational Matrix Gaussian (VMG) is Louizos and Welling [2016]. Functional Bayesian Neural Networks (FBNN) is Sun et al. [2019]. Probabilistic Backpropagation Matrix Variate Gaussian (PBP_MV) is Sun et al. [2017]. Deterministic VI (DVI) is Wu et al. [2019].

of the loss. We use rejection sampling, selecting only samples from a Gaussian distribution which fall under a threshold.

Our new estimate of the loss is biased (because we are not sampling from the distribution used to compute the KL divergence) but has lower variance (because only samples near the mean are used).

In Figure 10 we show that the truncated models to outperform 'correct' MFVI with standard deviations initialized slightly too high (we used $\sigma = 0.12$). This is despite the fact that we are using a biased estimator. This supports the hypothesis that MFVI training is hamstrung by high gradient variance.

Moreover, the smaller the number of samples, the higher the variance of the estimator will be, and the bigger a problem we might expect variance to be for training. Indeed, we show that the effect of truncation is smaller for larger numbers of samples. This suggests that estimating the gradient of the loss function for MFVI is hampered by sampling far from the mean, and that this effect is linked to the variance of estimates of the gradient.