Uncertainty in Neural Networks: Bayesian Ensembling

Tim Pearce
University of Cambridge
The Alan Turing Institute

Mohamed Zaki
University of Cambridge

Alexandra Brintrup
University of Cambridge

Andy Neely
University of Cambridge

Abstract

Understanding the uncertainty of a neural network's (NN) predictions is essential for many applications. The Bayesian framework provides a principled approach to this, however applying it to NNs is challenging due to the large number of parameters and data. Ensembling NNs provides a practical and scalable method for uncertainty quantification. Its drawback is that its justification is heuristic rather than Bayesian. In this work we propose one modification to the usual ensembling process, that does result in Bayesian behaviour: regularising parameters about values drawn from a prior distribution. Hence, we present an easily implementable, scalable technique for performing approximate Bayesian inference in NNs.

1 INTRODUCTION

By many measures neural networks (NNs) are the current dominant force within machine learning, however, they are not probabilistic in nature, which makes understanding the uncertainty of their predictions a challenge. This is vital for many real-world applications: if a healthcare system recommends a particular procedure, it is little comfort for the patient and doctor to know that, on average, the system's decisions are of good quality. Rather, the uncertainty of that individual prediction is important. Uncertainty is also useful in auxiliary ways; for exploration in reinforcement learning (RL), active learning and in identifying out-of-distribution instances (Sünderhauf et al., 2018).

Training a model to output uncertainty estimates cannot directly be framed as a supervised learning task. That is, there is no obvious uncertainty 'label' to assign to individual inputs. Rather, a model must be set up in such a way that it can infer this itself.

A principled approach is provided by the Bayesian framework, which models uncertainty in model parameters, enabling output of a *predictive distribution* as opposed to a point estimate. Bayesian Neural Networks (BNNs) are NNs over which Bayesian inference is performed (MacKay, 1992). Whilst appealing, parameters in modern NNs can be in the order of millions, trained over massive datasets, and this renders many Bayesian inference techniques that work well in small-scale settings infeasible.

If one half of the challenge is in running Bayesian inference at such scale, the other half, less discussed, is in limiting its impact on the current use of NNs in

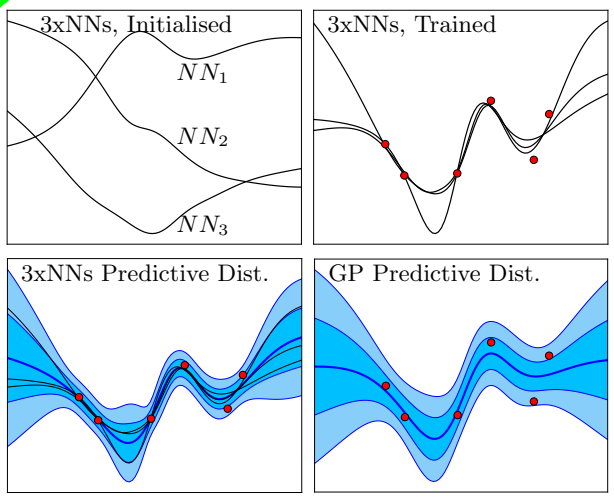


Figure 1: An ensemble of three NNs, starting from different initialisations and trained with the proposed loss (eq. 11), produce a predictive distribution approximating that of a GP. The approximation improves with increasing NN width and number of NNs.

Preliminary work.

practice (Gal, 2016) [p. 14]. Libraries like Tensorflow and PyTorch are becoming embedded in the machine learning community; it seems unlikely that methods straying from these frameworks will be widely adopted.

Ensembling provides one way to estimate uncertainty: it aggregates the estimates of multiple individual NNs, trained from different initialisations and sometimes on noisy versions of the training data. The variance of the ensemble's predictions may be interpreted as its uncertainty. The intuition is simple: predictions converge to similar results where data has been observed, and will be diverse elsewhere. The chief attraction is that the method scales well to large parameter and data settings, and each individual NN is implemented in precisely the usual way.

The downside is that ensembling, in its usual form, is not Bayesian. Despite empirical success from Tibshirani (1996), Osband et al. (2016), and Lakshminarayanan et al. (2017), it has gained little traction in the uncertainty in deep learning community. Gal (2016), who provides an authoritative text on the matter, says ensembling "*cannot technically be considered as approximate inference in BNNs*" [p. 27]. Whilst we will show that the exemplum subsequently provided was ill-founded (section 5.1.1), the sentiment holds.

We address this critical weakness by proposing one modification to the usual ensembling process that does produce Bayesian behaviour - **instead of regularising parameters about zero, they are regularised about values drawn from a prior distribution**. This leverages a little known inference method, randomised MAP sampling (section 2.2). Figure 1 illustrates our method, which we name *anchored ensembling* as each NN is regularised, or 'anchored', about its initialisation values.

Previous work applying similar ideas to NNs conducted limited analysis into its justification, directly applying schemes that were consistent for linear regression (Lu and Van Roy, 2017). We show that, in fact, the randomised MAP sampling scheme commonly proposed for linear regression *cannot* be applied to NNs. We propose an alternative scheme that is consistent for wide NNs (sections 3 & 4). On regression benchmarking datasets, anchored ensembles convincingly outperform current state-of-the-art methods in cases where epistemic model uncertainty is of primary importance (section 5), approaching the performance limit achievable with the equivalent Gaussian Process (GP).

Perhaps most appealing is our method's practicality, requiring a small number of NNs (say, 5 to 10), implemented in the usual way (using any modern deep learning library), with only a slightly modified regularised loss function. For reasonably wide NNs (50-100 nodes sufficed in experiments), this results in low bias compared to exact Bayesian inference methods.

2 BACKGROUND

2.1 Bayesian Inference in Neural Networks

A variety of methods have been developed to perform Bayesian inference in NNs. Recently variational inference (VI) has received much attention (Graves, 2011; Hernández-Lobato and Adams, 2015). A disadvantage of mean-field VI (a common form) is that it does not maintain correlations between parameters, and its appropriateness for NNs has been questioned (Ritter et al., 2018; Osband et al., 2018). Dropout was shown to perform mean-field VI with Bernoulli approximating distributions - *MC Dropout*, (Gal and Ghahramani, 2015), although further examinations have queried this (Osband et al., 2018; Hron et al., 2018; Melis et al., 2018).

Other inference methods include: Hamiltonian Monte Carlo (HMC), a MCMC variant which provides 'gold standard' inference but at limited scalability (Neal, 1997); Laplace approximations of the posterior requiring computation of the Hessian (or approximations of) (Ritter et al., 2018); ensembling in conjunction with early stopping (Duvenaud and Adams, 2016); finally, though not Bayesian, borderline 'out-of-distribution' samples of high variance can be synthesised and added to the training dataset (Lee et al., 2017).

Famously, BNNs of infinite width converge to GPs (Neal, 1997). Analytical kernels have been derived for single-layer NNs with certain activation functions, including Error Function (ERF) (a type of sigmoid), Radial Basis Function (RBF) (Williams, 1996), Rectified Linear Unit (ReLU) (Cho and Saul, 2009), and leaky ReLU (Tsuchida et al., 2018). **Their practicality is limited by their poor scalability relative to NNs**: $\mathcal{O}(N^3)$ for matrix inversion, $\mathcal{O}(N^2)$ for kernel computation. However, in small scale problems these GP kernels provide an opportunity to do *exact* inference, and are thereby valuable as a comparison to scalable methods in wide BNNs. Deep BNNs do not have analytical GP kernels, requiring numerical computation as in Lee et al. (2018).

2.2 Randomised MAP Sampling

Recent work in the Bayesian community, and independently in the RL community, has begun to explore an approach to Bayesian inference that will be novel to many readers. Roughly speaking, it exploits the fact that adding a regularisation term to a loss function returns maximum a posteriori (MAP) estimates

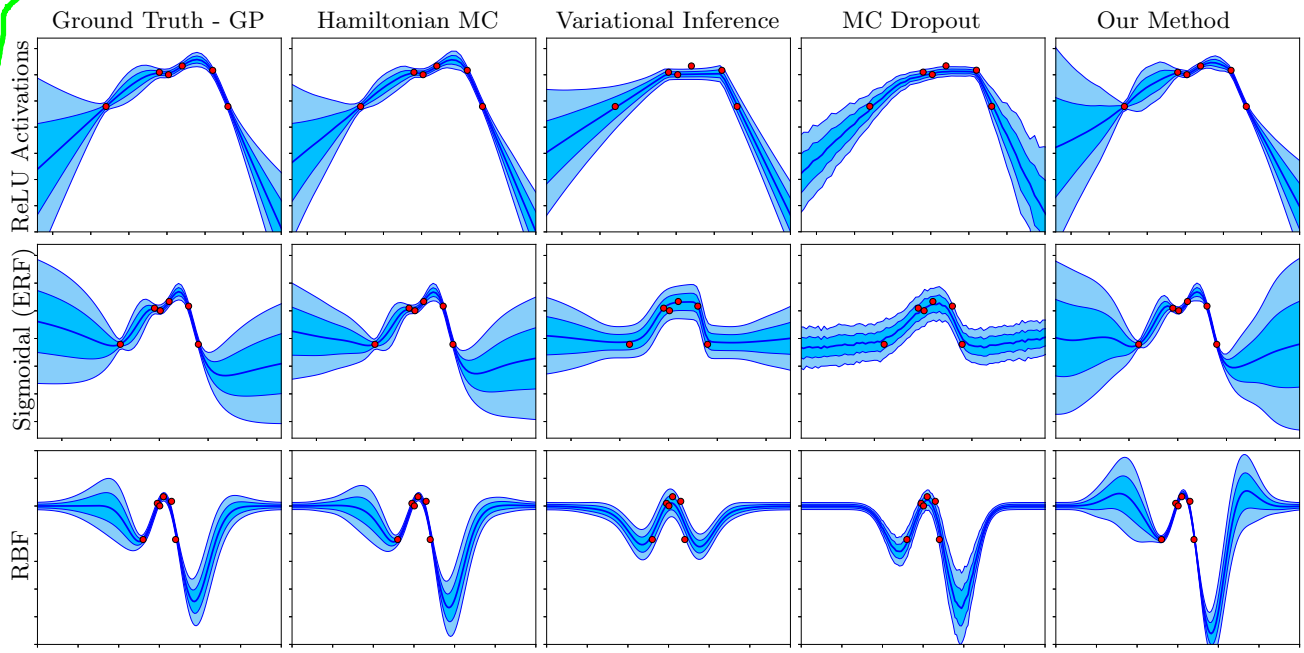


Figure 2: Predictive Distributions Produced by Various Inference Methods (Columns) with Varying Activation Functions (Rows) in Wide Single-Layer NNs: e.g. Bottom right is a RBF NN with inference by our method.

of parameters. Injecting noise into this loss, either to targets or regularisation term, and sampling repeatedly (i.e. ensembling), produces a distribution of MAP solutions mimicking that of the true posterior. This can be an efficient method to sample from high-dimensional posteriors (Bardsley et al., 2014).

Whilst it is straightforward to select the noise distribution that produces exact inference in linear regression models, there is difficulty in transferring this idea to more complex settings, such as NNs. Directly applying the noise distribution from the linear case to NNs has had some empirical success, despite not reproducing the true posterior (Lu and Van Roy, 2017; Osband et al., 2018). A more accurate, though more computationally demanding solution, is to wrap the optimisation step into an MCMC procedure (Bardsley, 2012; Bardsley et al., 2014).

Variants of this technique have been published under several names including randomise-then-optimise, randomised prior functions, and ensemble sampling. We refer to this family of procedures as *randomised MAP sampling*.

2.3 Regression Philosophy

Through this work we will assume a regression model of the form,

$$y = f(\mathbf{x}) + \epsilon, \tag{1}$$

for some function $f : \mathbb{R}^D \to \mathbb{R}$, a D-dimensional input vector, $\mathbf{x} \in \mathbb{R}^D$, and scalar target, $y \in \mathbb{R}$. Multiple inputs and outputs are denoted by $\mathbf{X}$ and $\mathbf{y}$, for a total of N training data points. A single test data point is denoted, $\mathbf{x}^*$. In this work we will only consider the case of homoskedastic, normally distributed errors, $\epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2)$. We will assume that the noise term has already been estimated, $\hat{\sigma}_\epsilon^2$ (this is relatively simple in practice). Hatted symbols signify estimates.

The uncertainty of y, quantitatively defined as its variance, σ_y^2, decomposes into two components corresponding to terms in eq. 1,

$$\sigma_y^2 = \sigma_{model}^2 + \sigma_\epsilon^2. \tag{2}$$

Epistemic or model uncertainty, σ_{model}^2, is the uncertainty in f, whilst aleatoric or data noise uncertainty, σ_ϵ^2, is that arising from ϵ. Pearce et al. (2018) provide a full discussion.

2.4 Regularised Neural Networks

We will be considering single-layer NNs of the form,

$$\hat{y} = \psi(\mathbf{x}\mathbf{W}_1 + \mathbf{b}_1)\mathbf{W}_2, \tag{3}$$

for some non-linearity, ψ, and parameters $\mathbf{W}_1, \mathbf{b}_1, \mathbf{W}_2$. Hidden width is given by, H, so $\mathbf{W}_1$ is of dimension DxH, $\mathbf{b}_1$ a vector of length H, and $\mathbf{W}_2$ dimension Hx1. Double-layer networks are built in the usual recursive way by introducing $\mathbf{b}_2, \mathbf{W}_3$, and deeper networks similarly. Note we have dropped the final bias to slightly simplify our analysis.

NNs are usually trained with a regularised loss,

$$Loss_{regularise} = \frac{1}{N}||\mathbf{y}_i - \hat{\mathbf{y}}||_2^2 + \frac{1}{N}||\mathbf{\Gamma}^{1/2}\boldsymbol{\theta}||_2^2, \tag{4}$$

where $\boldsymbol{\theta}$ is a flattened vector of all parameter values. Readers may be more familiar with a regularisation term, $\frac{\lambda}{N}||\boldsymbol{\theta}||^2$, but the above form allows flexibility of selection of λ's for different parameter layers. These are combined in $\mathbf{\Gamma}$, which is a square diagonal matrix. If $\mathbf{\Gamma} = \lambda \cdot \mathbb{I}$, where $\mathbb{I}$ is the identity matrix, then the forms are equivalent. When $\mathbf{\Gamma} = 0$, there is no regularisation (*unconstrained*).

Parameters minimising this loss can be interpereted from a Bayesian persepective as MAP estimates with a normal prior centered at zero (MacKay, 2005), and the k^{th} diagonal element of $\mathbf{\Gamma}$ the ratio of data noise to prior variance for parameter θ_k,

$$diag(\mathbf{\Gamma})_k = \frac{\sigma_\epsilon^2}{\sigma_{prior_k}^2}, \tag{5}$$

with normal prior, $P(\theta_k) = \mathcal{N}(0, \sigma_{prior_k}^2)$.

For a single NN, regularisation prevents overfitting, with eq. 4 widely used. But when ensembling for uncertainty, regularisation produces poor results since it encourages all NNs in the ensemble to the same single solution. Meanwhile, the unconstrained form is also inappropriate: although it produces diversity, **no notion of prior is maintained, and hence it is not Bayesian**. Figure 3 demonstrates this dilemma and shows how our proposed method provides a solution.

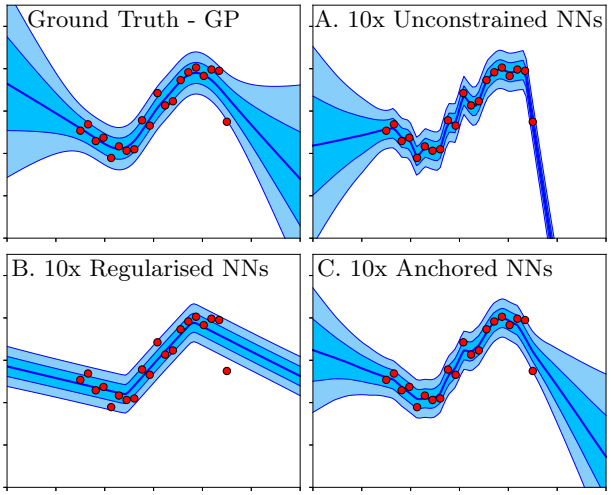


Figure 3: The Ensembling Regularisation Dilemma: regularisation reduces diversity in the ensemble (B), however removing it overfits the data (A), anchored ensembling provides a solution (C).

3 RANDOMISED ANCHORED MAP SAMPLING

In this section we present a general scheme for Bayesian inference which is a new variant of randomised MAP sampling (section 2.2). Two aspects distinguish it from prior work: first we consider the general case of a normally distributed prior and likelihood, without assuming a linear model, secondly we propose adding noise only to the regularisation term and not targets. We term this Bayesian scheme *randomised anchored MAP sampling*, and *anchored ensembles* for short.

Consider multivariate normal prior and (normalised) likelihood, $\mathcal{N}(\boldsymbol{\mu}_{prior}, \mathbf{\Sigma}_{prior}), \mathcal{N}(\boldsymbol{\mu}_{like}, \mathbf{\Sigma}_{like})$. The posterior, also multivariate normal, is given by Bayes rule, $\mathcal{N}(\boldsymbol{\mu}_{post}, \mathbf{\Sigma}_{post}) \propto \mathcal{N}(\boldsymbol{\mu}_{prior}, \mathbf{\Sigma}_{prior}) \cdot \mathcal{N}(\boldsymbol{\mu}_{like}, \mathbf{\Sigma}_{like})$. The MAP solution is simply $\boldsymbol{\mu}_{post}$. In general,

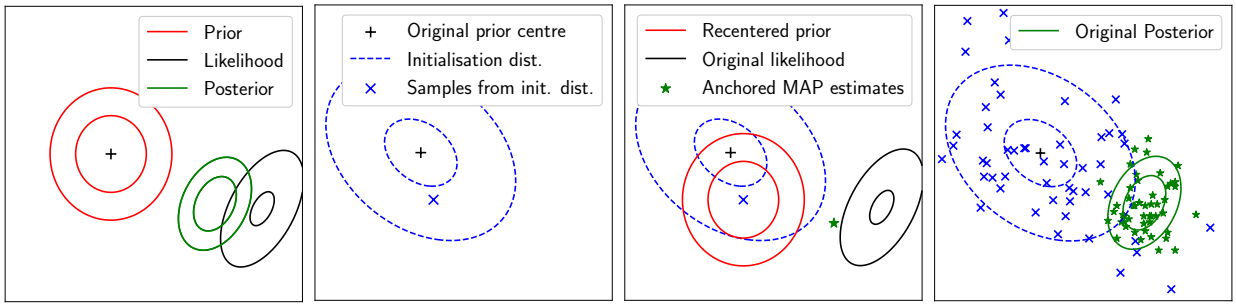
$$\boldsymbol{\mu}_{post} = (\mathbf{\Sigma}_{like}^{-1} + \mathbf{\Sigma}_{prior}^{-1})^{-1}(\mathbf{\Sigma}_{like}^{-1}\boldsymbol{\mu}_{like} + \mathbf{\Sigma}_{prior}^{-1}\boldsymbol{\mu}_{prior}). \tag{6}$$

For linear regression, $y = \boldsymbol{\theta}^T\mathbf{x} + \epsilon$, this becomes,

$$\boldsymbol{\mu}_{post} = (\frac{1}{\sigma_\epsilon^2}\mathbf{X}^T\mathbf{X} + \mathbf{\Sigma}_{prior}^{-1})^{-1}(\frac{1}{\sigma_\epsilon^2}\mathbf{X}^T\mathbf{y} + \mathbf{\Sigma}_{prior}^{-1}\boldsymbol{\mu}_{prior}). \tag{7}$$

In randomised MAP sampling we are interested in injecting noise so that $\mathbb{V}ar[\boldsymbol{\mu}_{post}] = \mathbf{\Sigma}_{post}$. From eq. 7 we see two feasible candidates that could be turned into noisy random variables: $\mathbf{y}$ or $\boldsymbol{\mu}_{prior}$. Previous work injected noise into both these terms (Lu and Van Roy, 2017; Osband et al., 2018).

However, for models other than linear regression, it may not be possible to move from eq. 6 to an analytical form containing $\mathbf{y}$ as in eq. 7. Furthermore, as in the case of a NN, different parameters within a model



Analytical Bayesian Inference

Step 1: Set initialisation dist. as $\mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$ from eq. 9 & 10. Sample $\boldsymbol{\theta}_0$ once.

Step 2: Return $\boldsymbol{\mu}_{post}^{MAP}(\boldsymbol{\theta}_0)$ **Step 3:** Repeat steps 1 & 2 from eq. 8 with $\boldsymbol{\theta}_0$ sampled from step one. to recover original posterior.

Figure 4: Demonstration of Randomised Anchored MAP Sampling in a 2-D Parameter Space.

may require different noise distributions to be added to $\mathbf{y}$, which is impossible given they share a single set of targets. Hence, **adding noise to targets is not viable for a NN**.

If instead $\boldsymbol{\mu}_{prior}$ is chosen as the sole noise source, this problem is avoided. In order to inject this noise, let us replace $\boldsymbol{\mu}_{prior}$ with some noisy random variable, $\boldsymbol{\theta}_0$, and denote $\boldsymbol{\mu}_{post}^{MAP}(\boldsymbol{\theta}_0)$ the resulting MAP estimate,

$$\boldsymbol{\mu}_{post}^{MAP}(\boldsymbol{\theta}_0) = (\boldsymbol{\Sigma}_{like}^{-1} + \boldsymbol{\Sigma}_{prior}^{-1})^{-1}(\boldsymbol{\Sigma}_{like}^{-1}\boldsymbol{\mu}_{like} + \boldsymbol{\Sigma}_{prior}^{-1}\boldsymbol{\theta}_0). \tag{8}$$

Derivation of the noise distribution required for $\boldsymbol{\theta}_0$ is given in appendix A, found by setting $\mathbb{V}ar[\boldsymbol{\mu}_{post}^{MAP}(\boldsymbol{\theta}_0)] = \boldsymbol{\Sigma}_{post}$. We find $\boldsymbol{\theta}_0 \sim \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$,

$$\boldsymbol{\mu}_0 = \boldsymbol{\mu}_{prior} \tag{9}$$

$$\boldsymbol{\Sigma}_0 = \boldsymbol{\Sigma}_{prior} + \boldsymbol{\Sigma}_{prior}^2\boldsymbol{\Sigma}_{like}^{-1}. \tag{10}$$

Figure 4 provides a visual demonstration of the algorithm over a 2-D parameters space.

4 ANCHORED ENSEMBLES OF NEURAL NETWORKS

Although the previous section's result is of interest, evaluating eq. 10 requires knowing the likelihood covariance, $\boldsymbol{\Sigma}_{like}$. Estimating this for a NN is far from simple: NNs are unidentifiable, their likelihood variances and correlations vary greatly across parameters, and $\boldsymbol{\Sigma}_{like}$ is not static, shifting during training. The only reasonable approximations are conditional on other parameter values already being fixed, which leads to a circularity in needing to know the values of the parameters before they are initialised.

This impasse can be solved in a surprising way. From eq. 10 we see that $diag(\boldsymbol{\Sigma}_0) \geq diag(\boldsymbol{\Sigma}_{prior})$. In fact, we find that with increasing width, H, the term $\boldsymbol{\Sigma}_{prior}^2\boldsymbol{\Sigma}_{like}^{-1}$ tends to a zero matrix. Therefore using this lower bound and setting $\boldsymbol{\Sigma}_0 = \boldsymbol{\Sigma}_{prior}$ gives good empirical results for wide NNs. Appendix B provides detailed analysis to support this. We offer a summary below.

It is usual to scale prior covariance in BNNs in a similar manner to initialisation variances in NNs (e.g. the Xavier scheme) - according to $1/H$ (Neal, 1997). This means the term of interest, $\boldsymbol{\Sigma}_{prior}^2\boldsymbol{\Sigma}_{like}^{-1} \propto \frac{1}{H^2}\boldsymbol{\Sigma}_{like}^{-1}$, which clearly decreases with H (appendix B.1).

Furthermore, increasing H creates more parameters and hence a higher probability of strong correlations amongst them (appendix B.2). In a similar way to multicollinearity in linear regression, this has the effect of increasing the magnitude of $\boldsymbol{\Sigma}_{like}$. Hence $\boldsymbol{\Sigma}_{like}^{-1}$ decreases. Both these results suggest, $\lim_{H\to\infty} \boldsymbol{\Sigma}_{prior}^2\boldsymbol{\Sigma}_{like}^{-1} \to 0$.

Stepping back, we note $\boldsymbol{\Sigma}_0 \approx \boldsymbol{\Sigma}_{prior}$ is only true in the case the posterior is dominated by the prior distribution rather than the likelihood. This occurs in BNNs because the role of priors is slightly abused as a source of regularisation in an over-paramatised model.

This observation is significant as it allows us to **relax our assumption that the prior and likelihood be normally distributed**. Instead, we can say that our method is valid provided the posterior is dominated by the prior.

A surprisingly simple result remains: a wide NN, minimising the following loss function,

Algorithm 1 Implementing Anchored Ensembles

Input: Training data, $\mathbf{X}$ & $\mathbf{y}$, test data point, $\mathbf{x}^*$, prior mean and covariance, $\boldsymbol{\mu}_{prior}$, $\boldsymbol{\Sigma}_{prior}$, ensemble size, M, data noise variance estimate, $\hat{\sigma}^2_\epsilon$.
Output: Estimate of predictive distribution mean and variance, $\hat{y}$, $\hat{\sigma}^2_y$.

 # Set regularisation matrix, eq. 5
$\boldsymbol{\Gamma} \Leftarrow \hat{\sigma}^2_\epsilon \boldsymbol{\Sigma}^{-1}_{prior}$

 # Train ensemble
for $j = 1$ **to** M
  $\boldsymbol{\mu}_0 \Leftarrow \boldsymbol{\mu}_{prior}, \boldsymbol{\Sigma}_0 \Leftarrow \boldsymbol{\Sigma}_{prior}$
  Initialise NN_j from $\mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$
  $\boldsymbol{\theta}_{j,0} \Leftarrow$ initialised parameters
  $NN_j.train(\mathbf{X}, \mathbf{y}, \boldsymbol{\Gamma}, \boldsymbol{\theta}_{j,0})$, loss in eq. 11
end for

 # Predict with ensemble
for $j = 1$ **to** M
  $\hat{y}_j \Leftarrow NN_j.predict(\mathbf{x}^*)$
end for

$\hat{y} = \frac{1}{M}\sum_{j=1}^{M} \hat{y}_j$, *# Mean prediction*
$\hat{\sigma}^2_{model} = \frac{1}{M-1}\sum_{j=1}^{M}(\hat{y}_j - \hat{y})^2$ *# Epistemic var.*
$\hat{\sigma}^2_y = \hat{\sigma}^2_{model} + \hat{\sigma}^2_\epsilon$ *# Total var. from eq. 2*

return $\hat{y}$, $\hat{\sigma}^2_y$

$$Loss_{anchor,j} = \frac{1}{N}||\mathbf{y} - \hat{\mathbf{y}}_j||_2^2 + \frac{1}{N}||\boldsymbol{\Gamma}^{1/2}(\boldsymbol{\theta}_j - \boldsymbol{\theta}_{0,j})||_2^2, \tag{11}$$

$$\text{where} \quad \boldsymbol{\theta}_{0,j} \sim \mathcal{N}(\boldsymbol{\mu}_{prior}, \boldsymbol{\Sigma}_{prior}) \tag{12}$$

produces a single posterior sample, where different $\boldsymbol{\theta}_{0,j}$ is drawn for each ensemble member, NN_j. Algorithm 1 demonstrates the process of training and predicting with anchored ensembles of NNs, including how to combine the ensemble's estimates. The training step is done in exactly the usual way, minimising the objective in eq. 11 with stochastic gradient descent (SGD), using any optimiser and no early stopping. Note that although not strictly required, it is convenient to use $\boldsymbol{\theta}_{0,j}$ for initialisations as well as regularisation points.

4.1 Ensemble Size

How many NNs should be included in an anchored ensemble? If each NN is a single posterior sample, an inordinate number would be required to capture the true posterior parameter distributions. But the parameter distributions themselves are of little interest in the context of a NN. It is the predictive distribution that is of sole interest. In this way, we move from doing inference in parameter space to output space.

Given that each NN provides an independent sample from a posterior predictive distribution, a relatively small number of NNs can give a good approximation. More will provide better accuracy, but at increased computational cost. The marginal accuracy improvement per NN decreases in the usual manner of variance and sample size - for a Gaussian predictive distribution, the standard error of mean and variance decrease according to $1/M$, where M is ensemble size. As a rule of thumb, we suggest an ensemble size of 5-10. This number does not increase with dimensionality of input or output.

4.2 NN Width

Given anchored ensembles require a wide NN, the second question is: how large should H be for the method to be valid? Again there is a trade-off; wider NNs should offer increasingly accurate inference, but require more computation. Empirically speaking, in section 5, anchored ensembles surpassed state-of-the-art results with NNs of 50-100 nodes, suggesting this width is sufficient for the claims to hold. Figure 6 shows diminishing returns in inference quality as width increases, with a NN of $1,024$ nodes achieving close to the performance limit.

5 RESULTS

Code for the all experiments in this section is available online (github.com/TeaPearce). Hyperparameter details are given in appendix E.

5.1 Visual Comparison of Inference Methods

In figure 2 we compare popular Bayesian inference methods in single-layer NNs with our method. We used several non-linearities for which analytical GP kernels exist - ReLU, RBF, and ERF[1]. Leaky ReLU is included in appendix figure 10. The same six data points were used for all plots, NNs had 100 hidden nodes, hyperparameters for priors and data noise were shared by all methods.

GP and HMC produce 'gold standard' Bayesian inference (although are not scalable). Hence, we judge the remaining methods, which are scalable approximations, to them. Because the NN is wide relative to the problem, HMC closely approximates the GP.

We implemented mean-field VI with Gaussian approximating 'q-distributions'. VI captures the geometry

[1]No analytical GP kernel exists for the more common sigmoidal TanH or logistic functions. ERF is similar to TanH.

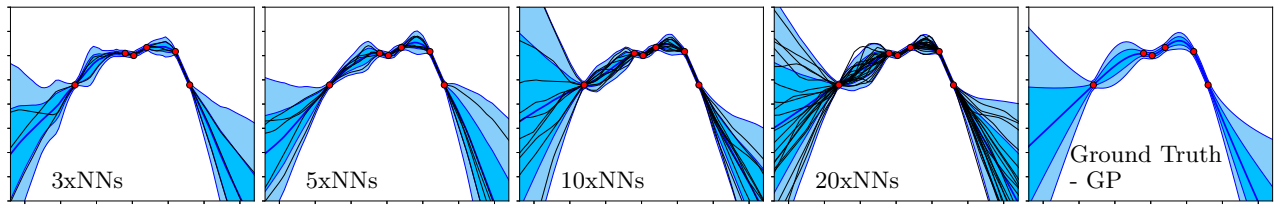


Figure 5: As the Number of NNs in an Anchored Ensemble Increases, the Predictive Distribution Roughly Converges to that of the Equivalent GP.

of extrapolations reasonably well, but because it does not account for correlations between parameters, fails to capture uncertainty in interpolations. MC Dropout performs similarly, though also fails to extrapolate uncertainty for the ERF case. This is because it uses the less flexible Bernoulli q-distribution at the *output* of nodes (as opposed to modelling the parameters themselves). We believe this would be the case for any odd non-linearity in a single-layer NN.

Our method, anchored ensembling, used ten NNs per ensemble, and hence training takes ten times that of a single NN. However predictions are faster than other methods, which generally perform forward passes using more than ten parameter samples. The predictive distributions, although somewhat wavy, are good approximations of GP and HMC, both for extrapolations and interpolations, though the variance tends to be slightly too large.

5.1.1 Erroneous Argument

The argument referred to in section 1 claimed that an ensemble of RBF NNs would output zero with high confidence when predicting far from the training data, and this would not be the case for the equivalent RBF GP which was the squared exponential (SE) kernel (Gal, 2016) [p. 27]. However, the RBF GP is *not* the SE kernel except in the special case of infinite variance priors (Williams, 1996). Figure 2, bottom left, shows the *actual* RBF GP for the case of finite variance. In fact the GP outputs zero with high confidence far from the data, as do all methods.

5.2 Visual Convergence Test

Figure 5 shows the predictive distribution of an anchored ensemble with increasing numbers of single-layer NNs compared to exact ReLU GP inference. The distribution grows smoother and increasingly similar to that of the GP, however even with 20 NNs there is a slight residual difference between the two: our method's extrapolations are flatter and of larger variance. This suggests the posterior found by anchored ensembles contains some slight bias compared to the true posterior. We believe this is caused by parameter likelihoods changing suddenly as data points cross the elbow of ReLU nodes. Appendix figure 9 shows this.

5.3 Boston Convergence Test

We quantitatively compared the predictive distributions of anchored ensembles to exact ReLU GP inference on the Boston dataset as in figure 6. Varying both the width of the NN, and number of NNs in the ensemble, we measured the Kullback-Leibler (KL) divergence between the two. Training was done on 50% of the data, testing on the other 50%. Results were averaged over ten runs. The 'ideal' line shows the metric

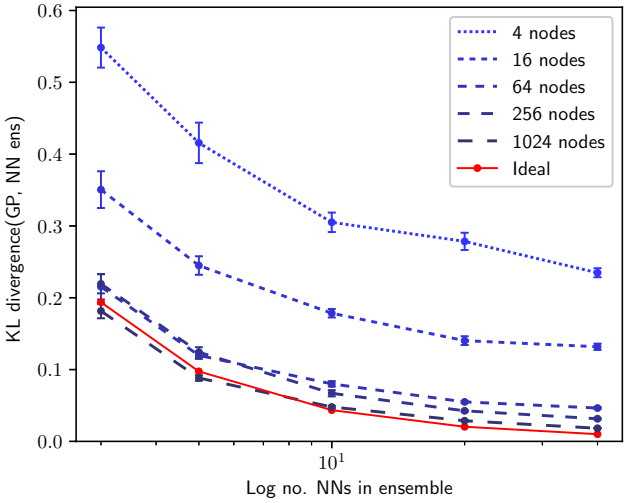


Figure 6: Predictive Distribution of Anchored Ensemble Converges to a ReLU GP's as Width of the NN, and Number in Ensemble, Increase. Boston Dataset. Error Bars ±1 Standard Error.

Table 1: NLL Regression Benchmark Results. See Appendix C for RMSE and Variants of Our Method. Mean ±1 Standard Error.

	$\hat{\sigma}_\epsilon^2$	Deep Ens. *State-Of-Art*	Anch. Ens. *Our Method*	ReLU GP[1] *Gold Standard*
High Epistemic Uncertainty				
Energy	1e-7	1.38 ± 0.22	**0.96 ± 0.13**	0.86 ± 0.02
Naval	1e-7	-5.63 ± 0.05	**-7.17 ± 0.03**	-10.05 ± 0.02
Yacht	1e-7	1.18 ± 0.21	**0.37 ± 0.08**	0.49 ± 0.07
Equal Epistemic & Aleatoric Uncertainty				
Kin8nm	0.02	-1.20 ± 0.02	**-1.09 ± 0.01**	-1.22 ± 0.01
Power	0.05	**2.79 ± 0.04**	2.83 ± 0.01	2.80 ± 0.01
Concrete	0.05	3.06 ± 0.18	**2.97 ± 0.02**	2.96 ± 0.02
Boston	0.08	**2.41 ± 0.25**	2.52 ± 0.05	2.45 ± 0.05
High Aleatoric Uncertainty				
Protein	0.5	**2.83 ± 0.02**	2.89 ± 0.01	*2.88 ± 0.00
Wine	0.5	**0.94 ± 0.12**	**0.95 ± 0.01**	0.92 ± 0.01
Song	0.7	**3.35 ± NA**	3.60 ± NA	**3.62 ± NA

[1] For reference only (not a scalable method). * Trained on 10,000 rows of data. ** Trained on 20,000 rows of data, tested on 5,000 data points.

when posterior samples from the GP itself, rather than anchored NNs, were used.

Increasing both NN width and number of NNs in the ensemble decreases KL divergence, with little extra performance gained by widening the NN beyond 64 nodes, and increasing the ensemble size beyond 10. Close to ideal performance is achieved by the 1,024 node NN, with very low residual bias compared to that observed in section 5.2 - this is possibly due to a larger dataset reducing the impact of piece-wise likelihoods.

5.4 Benchmarking

We benchmarked our method on the regression experiments introduced by Hernandez-Lobato & Adams (2015), assessing negative log likelihood (NLL) across ten datasets of varying size and dimensionality. Single-layer NNs of 50 nodes were used, experiments repeated 20 times with random train/test splits of 90%/10%. The larger Protein and Song datasets allow 100 node NNs, and were repeated five and one time respectively.

Since relatively wide single-layer NNs are used, we compared our method to exact inference using the ReLU GP, which has not been implemented before. We argue this gives the upper limit on performance for a single-layer NN assuming additive Gaussian homoskedastic noise. For Protein and Song datasets the GP could only be run on portions of the data.

We primarily compare to Deep Ensembles (Lakshminarayanan et al., 2017), which is the current state-of-the-art *scalable* method, having outperformed VI and MC Dropout. Deep Ensembles and our method both used five NNs per ensemble.

Table 1 shows results. Anchored ensembles performed slightly worse than the GP. This gap would decrease with increased NN width and ensemble number. ReLU GP performance was similar to reported HMC results [2] (Bui et al., 2016).

Ordering results according to the level of estimated data noise, $\hat{\sigma}_\epsilon^2$, shows a clear pattern - the GP and anchored ensembles excel in datasets with low data noise, whilst for high data noise, Deep Ensembles holds a slight advantage. This is because for low $\hat{\sigma}_\epsilon^2$, the main source of uncertainty is epistemic. For high $\hat{\sigma}_\epsilon^2$, aleatoric uncertainty is of primary importance (section 2.3). Both GP and anchored ensembles specialise in modelling epistemic uncertainty, and assume a constant value of aleatoric uncertainty. Deep Ensembles, on the other hand, aims to model both, with extra parameters dedicated to modelling heteroskedastic aleatoric uncertainty, and use of a more complex loss function. We emphasise that anchored ensembles is the significantly simpler model of the two.

In appendix C we provide results for variants of our method including increased ensemble size, two-layer NNs, the ERF GP kernel, and a single regularised NN with constant variance. The single NN produced surprisingly strong NLL results which we believe highlights a weakness in the benchmarking experiments - the prediction of high uncertainty for data drawn from a *new* distribution (out-of-distribution samples) is not directly tested. A critique is included in appendix F.

6 CONCLUSION

This paper considered a method to produce Bayesian behaviour in NN ensembles. We developed a new variant of randomised MAP sampling, showing how it can be applied to NNs. If NNs are sufficiently wide, each produces a sample from the posterior predictive distribution. The key attraction of the technique is its practicality, requiring 5 to 10 NNs trained in the usual manner, with parameters regularised around values drawn from a prior distribution.

Qualitatively it performs Bayesian inference more exactly than mean-field VI, both with a Gaussian q-distribution and Bernoulli q-distribution as in MC Dropout. This was demonstrated on four different activation functions.

On regression benchmarking experiments our method achieved state-of-the-art performance on tasks where epistemic uncertainty was of primary importance. We believe the method is equally applicable to classification; this should be formally verified in further work.

[2] HMC demonstrated its poor scalability taking three days to run on the Protein dataset.

Acknowledgements

The authors thank EPSRC for funding (EP/N509620/1), the Alan Turing Institute for accommodating the lead author during his work (TU/D/000016), and Microsoft for Azure credits. Personal thanks to Nicolas Anastassacos for early collaborations and edits, also to Ayman Boustati and Ian Osband for conversations and edits.

References

Bardsley, J. M. (2012). MCMC-based image reconstruction with uncertainty quantification. *SIAM Journal on Scientific Computing*, 34(3):1316–1332.

Bardsley, J. M., Solonen, A., Haario, H., and Laine, M. (2014). Randomize-Then-Optimize : A Method for Sampling from Posterior Distributions in Nonlinear Inverse Problems. *SIAM Journal on Scientific Computing*, 36(4).

Bui, T. D., Hernández-Lobato, D., Li, Y., Hernández-Lobato, J. M., and Turner, R. E. (2016). Deep Gaussian Processes for Regression using Approximate Expectation Propagation. In *Proceedings of the 33rd International Conference on Machine Learning*, volume 48.

Cho, Y. and Saul, L. K. (2009). Kernel Methods for Deep Learning. In *Advances in Neural Information Processing Systems 22 (NIPS 2009)*.

Duvenaud, D. and Adams, R. P. (2016). Early Stopping as Nonparametric Variational Inference. In *AI Stats*, volume 51, pages 1070–1077.

Gal, Y. (2016). *Uncertainty in Deep Learning*. PhD thesis.

Gal, Y. and Ghahramani, Z. (2015). Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *Proceedings of the 33rd International Conference on Machine Learning*.

Graves, A. (2011). Practical Variational Inference for Neural Networks. *Advances in Neural Information Processing Systems*, pages 1–9.

Hernández-Lobato, J. M. and Adams, R. P. (2015). Probabilistic Backpropagation for Scalable Learning of Bayesian Neural Networks. In *Proceedings of the 32nd International Conference on Machine Learning*.

Hron, J., Matthews, A. G. d. G., and Ghahramani, Z. (2018). Variational Bayesian dropout: pitfalls and fixes. In *Proceedings of the 35th International Conference on Machine Learning*.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. (2017). Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. In *31st Conference on Neural Information Processing Systems*.

Lee, J., Bahri, Y., Novak, R., Schoenholz, S. S., Pennington, J., and Sohl-dickstein, J. (2018). Deep neural networks as gaussian processes. In *ICLR*.

Lee, K., Lee, H., Lee, K., and Shin, J. (2017). Training Confidence-calibrated Classifiers for Detecting Out-of-Distribution Samples. pages 1–16.

Lu, X. and Van Roy, B. (2017). Ensemble Sampling. In *31st Conference on Neural Information Processing Systems*.

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

MacKay, D. J. C. (2005). *Information Theory, Inference, and Learning Algorithms David J.C. MacKay*, volume 100.

Melis, G., Blundell, C., Kocisky, T., Hermann, K. M., Dyer, C., and Blunsom, P. (2018). Pushing the bounds of dropout.

Neal, R. M. (1997). *Bayesian Learning for Neural Networks*. PhD thesis.

Osband, I., Aslanides, J., and Cassirer, A. (2018). Randomized Prior Functions for Deep Reinforcement Learning.

Osband, I., Blundell, C., Pritzel, A., and Van Roy, B. (2016). Deep Exploration via Bootstrapped DQN. In *Advances in Neural Information Processing Systems 29*.

Pearce, T., Zaki, M., Brintrup, A., and Neely, A. (2018). High-Quality Prediction Intervals for Deep Learning: A Distribution-Free, Ensembled Approach. In *Proceedings of the 35th International Conference on Machine Learning , ICML*, Stockholm.

Pedersen, M. S., Baxter, B., Templeton, B., Rishøj, C., Theobald, D. L., Hoegh-rasmussen, E., Casteel, G., Gao, J. B., Dedecius, K., Strim, K., Christiansen, L., Hansen, L. K., Wilkinson, L., He, L., Bar, M., Winther, O., Sakov, P., Hattinger, S., Petersen, K. B., and Rishø j, C. (2008). The Matrix Cookbook. *Matrix*, M:1–71.

Ritter, H., Botev, A., and Barber, D. (2018). A Scalable Laplace Approximation for Neural Networks. In *ICLR*, pages 1–15.

Sünderhauf, N., Brock, O., Scheirer, W., Hadsell, R., Fox, D., Leitner, J., Upcroft, B., Abbeel, P., Burgard, W., Milford, M., and Corke, P. (2018). The limits and potentials of deep learning for robotics. *The International Journal of Robotics Research*, 37:405–420.

Tibshirani, R. (1996). A Comparison of Some Error Estimates for Neural Network Models. *Neural Computation*, 8:152–163.

Tsuchida, R., Roosta-Khorasani, F., and Gallagher, M. (2018). Invariance of Weight Distributions in Rectified MLPs. In *Proceedings of the 35th International Conference on Machine Learning*.

Williams, C. K. I. (1996). Computing with infinite networks. In *Advances in Neural Information Processing Systems 9*.

Appendix to Uncertainty in Neural Networks: Bayesian Ensembling

A PROOFS

Theorem 1. *Assume that the joint likelihood of model parameters follows a multivariate normal distribution, that the prior is normally distributed, and that there exists a mechanism by which optimal MAP parameter estimates can be returned. The proposed anchored inference scheme provides a consistent estimator of the posterior.*

Proof. Consider prior and (normalised) likelihood distributions, both multivariate normal,

$$P(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\mu}_{prior}, \boldsymbol{\Sigma}_{prior}), \tag{13}$$

$$\frac{P(\mathcal{D}|\boldsymbol{\theta})}{P(\mathcal{D})} = \mathcal{N}(\boldsymbol{\mu}_{like}, \boldsymbol{\Sigma}_{like}), \tag{14}$$

with posterior calculated by Bayes rule,

$$P(\boldsymbol{\theta}|\mathcal{D}) = \frac{P(\mathcal{D}|\boldsymbol{\theta})P(\boldsymbol{\theta})}{P(\mathcal{D})}. \tag{15}$$

Standard Result 1: (§8.1.8, The Matrix Cookbook, 2008)
If both prior and likelihood are multivariate normal, the posterior is also normal and available in closed form,

$$P(\boldsymbol{\theta}|\mathcal{D}) = \mathcal{N}(\boldsymbol{\mu}_{post}, \boldsymbol{\Sigma}_{post}), \tag{16}$$

$$\boldsymbol{\Sigma}_{post} = (\boldsymbol{\Sigma}_{prior}^{-1} + \boldsymbol{\Sigma}_{like}^{-1})^{-1}, \tag{17}$$

$$\boldsymbol{\mu}_{post} = \boldsymbol{\Sigma}_{post}\boldsymbol{\Sigma}_{prior}^{-1}\boldsymbol{\mu}_{prior} + \boldsymbol{\Sigma}_{post}\boldsymbol{\Sigma}_{like}^{-1}\boldsymbol{\mu}_{like}. \tag{18}$$

We now introduce an initialising distribution which we enforce as multivariate normal,

$$\boldsymbol{\theta}_0 \sim P(\boldsymbol{\theta}_0) = \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0). \tag{19}$$

This is used as described in the text so that samples are taken from the initialising distribution, with a prior centered at each sample,

$$P(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}_0, \boldsymbol{\Sigma}_{prior}), \tag{20}$$

where $\boldsymbol{\Sigma}_{prior}$ is unchanged from eq. 13.

Denote $\boldsymbol{\mu}_{post}^{MAP}(\boldsymbol{\theta}_0)$ as the MAP estimates given this prior and the original likelihood from eq. 14.

We must show three things regarding $\boldsymbol{\mu}_{post}^{MAP}(\boldsymbol{\theta}_0)$:

- that its distribution is multivariate normal,

$$P(\boldsymbol{\mu}_{post}^{MAP}(\boldsymbol{\theta}_0)) = \mathcal{N}(\boldsymbol{\mu}_{post}^{MAP}, \boldsymbol{\Sigma}_{post}^{MAP}), \tag{21}$$

- that $\boldsymbol{\mu}_0$ & $\boldsymbol{\Sigma}_0$ can be selected in such a way that the mean of the distribution is equal to that of the original posterior

$$\boldsymbol{\mu}_{post}^{MAP} = \boldsymbol{\mu}_{post}, \tag{22}$$

- and also so that the covariance of the distribution is equal to that of the original posterior

$$\boldsymbol{\Sigma}_{post}^{MAP} = \boldsymbol{\Sigma}_{post}. \tag{23}$$

We make use of the following standard result.

Standard Result 2: (§8.1.4, The Matrix Cookbook 2008)
For a random variable, $\mathbf{x}$, normally distributed and with an affine transformation applied,

$$\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c), \tag{24}$$

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{b}, \tag{25}$$

$\mathbf{y}$ will also be normally distributed with parameters as follows,

$$\mathbf{y} \sim \mathcal{N}(\mathbf{A}\boldsymbol{\mu}_c + \mathbf{b}, \mathbf{A}\boldsymbol{\Sigma}_c\mathbf{A}^T). \tag{26}$$

Consider a single sample from the initialising distribution, $\boldsymbol{\theta}_0^*$, that is adopted by the prior as,

$$\mathcal{N}(\boldsymbol{\theta}_0^*, \boldsymbol{\Sigma}_{prior}). \tag{27}$$

Denote $\boldsymbol{\theta}_{post}^*$ as the MAP parameter estimate of the posterior formed by this prior and the likelihood in eq. 14. We have already seen that the posterior is also normally distributed, and its mean, which is also the MAP estimate, is given by combining eq. 17, 18 & 27,

$$\boldsymbol{\theta}_{post}^* = \ (\boldsymbol{\Sigma}_{prior}^{-1} + \boldsymbol{\Sigma}_{like}^{-1})^{-1}\boldsymbol{\Sigma}_{prior}^{-1}\boldsymbol{\theta}_0^* + (\boldsymbol{\Sigma}_{prior}^{-1} + \boldsymbol{\Sigma}_{like}^{-1})^{-1}\boldsymbol{\Sigma}_{like}^{-1}\boldsymbol{\mu}_{like}. \tag{28}$$

Defining for convenience,

$$\mathbf{A}_1 = (\boldsymbol{\Sigma}_{prior}^{-1} + \boldsymbol{\Sigma}_{like}^{-1})^{-1}\boldsymbol{\Sigma}_{prior}^{-1}, \tag{29}$$

$$\mathbf{b}_1 = (\boldsymbol{\Sigma}_{prior}^{-1} + \boldsymbol{\Sigma}_{like}^{-1})^{-1}\boldsymbol{\Sigma}_{like}^{-1}\boldsymbol{\mu}_{like}, \tag{30}$$

this becomes,

$$\boldsymbol{\theta}_{post}^* = \mathbf{A}_1\boldsymbol{\theta}_0^* + \mathbf{b}_1, \tag{31}$$

which is the same form as eq. 25. Hence, from Standard Result 2, if $\boldsymbol{\theta}_0^*$ is normally distributed, $\boldsymbol{\theta}_{post}^*$ **will also be normally distributed**.

Regarding the mean of $\boldsymbol{\theta}_{post}^*$, we have,

$$\mathbb{E}[\boldsymbol{\theta}_{post}^*] = \mathbb{E}[\mathbf{A}_1\boldsymbol{\theta}_0^* + \mathbf{b}_1] \tag{32}$$

$$= \mathbf{A}_1\mathbb{E}[\boldsymbol{\theta}_0^*] + \mathbf{b}_1. \tag{33}$$

By choosing the initialising distribution to be centered about the original prior,

$$\mathbb{E}[\boldsymbol{\theta}_0^*] = \mathbb{E}[\boldsymbol{\theta}_{prior}] = \boldsymbol{\mu}_{prior}, \tag{34}$$

we have,

$$\mathbb{E}[\boldsymbol{\theta}^*_{post}] = \boldsymbol{\Sigma}_{post}\boldsymbol{\Sigma}^{-1}_{prior}\boldsymbol{\mu}_{prior} + \boldsymbol{\Sigma}_{post}\boldsymbol{\Sigma}^{-1}_{like}\boldsymbol{\mu}_{like}, \tag{35}$$

which is consistent with eq. 18 and proves that **the means of the distributions are aligned**.

Finally we consider the variance of $\boldsymbol{\theta}^*_{post}$, which we wish to equal $\boldsymbol{\Sigma}_{post}$ by choosing $\boldsymbol{\Sigma}_0$. Using the form from eq. 31 we find,

$$\mathbb{V}ar[\boldsymbol{\theta}^*_{post}] = \mathbb{V}ar[\mathbf{A}_1\boldsymbol{\theta}^*_0 + \mathbf{b}_1] \tag{36}$$

$$= \mathbf{A}_1\mathbb{V}ar[\boldsymbol{\theta}^*_0]\mathbf{A}_1^T = \mathbf{A}_1\boldsymbol{\Sigma}_0\mathbf{A}_1^T \tag{37}$$

We require the following result,

$$\boldsymbol{\Sigma}_{post} = (\boldsymbol{\Sigma}^{-1}_{prior} + \boldsymbol{\Sigma}^{-1}_{like})^{-1} = \mathbf{A}_1\boldsymbol{\Sigma}_0\mathbf{A}_1^T. \tag{38}$$

Denoting, $\mathbf{C} := \boldsymbol{\Sigma}^{-1}_{prior} + \boldsymbol{\Sigma}^{-1}_{like}$, remembering that $\mathbf{A}_1$ is symmetric and rearranging,

$$\boldsymbol{\Sigma}_0 = \mathbf{A_1}^{-1}\mathbf{C}^{-1}\mathbf{A_1}^{-1} \tag{39}$$

$$= \mathbf{C}\boldsymbol{\Sigma}^{-1}_{prior}\mathbf{C}^{-1}\boldsymbol{\Sigma}^T_{prior}\mathbf{C^T}^{-1}, \tag{40}$$

which reduces to,

$$\boldsymbol{\Sigma}_0 = \boldsymbol{\Sigma}_{prior} + \boldsymbol{\Sigma}^2_{prior}\boldsymbol{\Sigma}^{-1}_{like}. \tag{41}$$

If $\boldsymbol{\Sigma}_{prior}$ is selected to be diagonal and isotropic, it can be replaced by $\lambda_{prior} \cdot \mathbb{I}$, which reduces to,

$$\boldsymbol{\Sigma}_0 = \lambda_{prior}\mathbb{I} + \lambda^2_{prior}\boldsymbol{\Sigma}^{-1}_{like}. \tag{42}$$

Therefore, **the variance of the distributions will be aligned**.

□

B LIKELIHOOD ANALYSIS

In this section we show that as the width of a BNN increases to infinity, $\mathbf{\Sigma}^2_{prior}\mathbf{\Sigma}^{-1}_{like}$ tends to a zero matrix. Our analysis tackles this from two perspectives. In section B.1 we first consider the case of a single parameter in a NN, assuming all other parameters are fixed (conditional likelihood). In section B.2 we then consider the joint likelihood of all parameters in a layer.

These two sections derive two distinct lines of reasoning that nevertheless lead to the same conclusion. Indeed the two motivations are complimentary in nature, and we believe both are reasons that anchored ensembles work in practice, since for finite H, both of these will only be partially true.

B.1 Single Parameter Conditional Likelihoods

B.1.1 Final Layer Weights

Here we consider the likelihood variance, $\sigma^2_{w,f}$, of a single weight, w_f, (assuming all other parameters fixed) in the final layer of a NN, with linear activation on the output and normally distributed errors of variance σ^2_ϵ. The variance at the input to the node in question is denoted $\sigma^2_{in,f}$.

Using results from simple linear regression,

$$\sigma^2_{w,f} = \frac{\sigma^2_\epsilon}{\sigma^2_{in,f}N}. \tag{43}$$

We will assume the usual scaling of prior variance, σ^2_{prior}, according to $1/H$, where H is number of nodes in a hidden layer (Neal, 1997). This is done so that activations maintain a constant variance through all layers of the NN, and assures that $\mathbb{E}[\sigma^2_{in,f}]$ is independent of H. Since σ^2_ϵ is a property of the data, we find,

$$\mathbb{E}[\sigma^2_{w,f}] \propto \frac{1}{N}, \tag{44}$$

which is independent of the width. This relationship is visualised in figure 7, where the likelihood distribution is plotted for two randomly selected parameters of each layer through a two-layer NN. Likelihood variance of the final weights, W_3, is constant for varying levels of H.

If we now consider the ratio of interest, $\mathbf{\Sigma}^2_{prior}\mathbf{\Sigma}^{-1}_{like}$, for a single final layer weight,

$$\mathbb{E}\Big[\frac{\sigma^4_{prior}}{\sigma^2_{w,f}}\Big] \propto \frac{N}{H^2}. \tag{45}$$

This means, $\lim_{H\to\infty}\mathbb{E}\Big[\frac{\sigma^4_{prior}}{\sigma^2_{w,f}}\Big] \to 0$.

B.1.2 Penultimate Layer Weights

We now consider the likelihood variance of a single weight in the second final layer of a NN, w_p, with likelihood variance, $\sigma^2_{w,p}$. Here w_f refers to the (now fixed) final layer weight corresponding to w_p, and $\sigma^2_{in,p}$ gives the input variance.

Our analysis follows that for the final layer weights. We ignore the non-linearity in between the second and final layers, noting that both ReLU and sigmoidal non-linearities would have the effect of further increasing the likelihood variance, helping our argument further.

If later layer parameters are fixed, we are effectively dividing the targets by w_f, which multiplies σ^2_ϵ by $1/w^2_f$,

$$\sigma^2_{w,p} = \frac{\sigma^2_\epsilon/w^2_f}{\sigma^2_{in,p}N}. \tag{46}$$

Again, if earlier layer priors in the NN are scaled to maintain constant variance, $\mathbb{E}[\sigma^2_{in,p}]$ is independent of H. We also scale w_f by $1/H$. Evaluating $\mathbb{E}[1/w_f^2]$ leads to a divergent improper integral, however we can bound it through Jensen's inequality,

$$\mathbb{E}[1/w_f^2] \geq 1/\mathbb{E}[w_f^2] = 1/\sigma^2_{w,f} = H, \tag{47}$$

$$\mathbb{E}[\sigma^2_{w,p}] \geq \frac{\sigma^2_\epsilon H}{\sigma^2_{in,p} N} \propto \frac{H}{N}. \tag{48}$$

We see evidence of this H/N relationship in figure 7, where the likelihood variance of the penultimate weights of a two-layer NN, W_2, increases with increasing H.

Our term of interest is then bounded by N/H^3,

$$\mathbb{E}\left[\frac{\sigma^4_{prior}}{\sigma^2_{w,p}}\right] \leq \frac{\sigma^2_{in,p} N}{\sigma^2_\epsilon H^3} \propto \frac{N}{H^3}. \tag{49}$$

This means, $\lim_{H\to\infty} \mathbb{E}\left[\frac{\sigma^4_{prior}}{\sigma^2_{w,p}}\right] \to 0$.

B.1.3 Other Parameters

We end our single parameter analysis here, without explicitly considering weights in earlier layers, where branching becomes a factor, making analysis increasingly complex. We hypothesise that the ratio of interest would remain of the order N/H^3, but may include other terms. We also have not analysed biases, though believe results similar to weights would be found. Plots from figure 7 support these hypotheses.

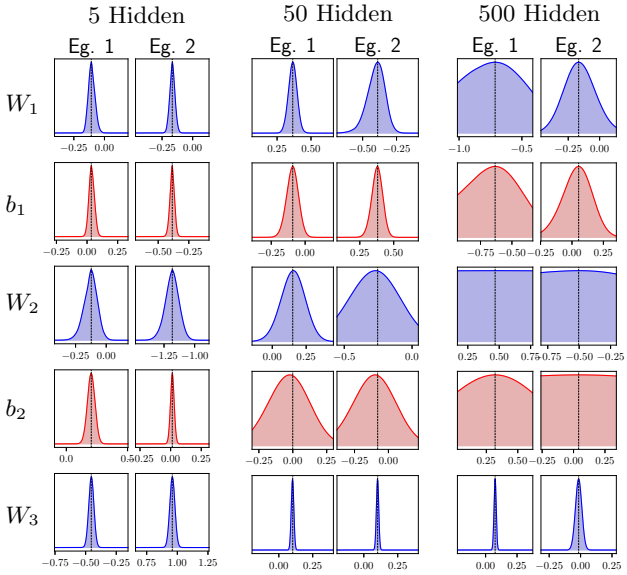


Figure 7: Likelihood of Parameters in a Two-Layer NN, Trained on Boston Dataset. As H increases, the likelihood variance increases for all parameter layers except for the final weight layer, W_3, for which it remains constant.

B.1.4 Implications for Full Covariance Matrix

In this section we've considered conditional likelihood variances. This analysis holds exactly for $\mathbf{\Sigma}^2_{prior}$ if priors assume no correlations between parameters, which is usual. The analysis also provides a lower bound on entries

in the full likelihood covariance matrix, $\mathbf{\Sigma}_{like}$, (and would only be exactly true in situations where no correlations exist between parameters). Therefore this provides insight into how $\mathbf{\Sigma}_{like}^{-1}$ changes with H and N.

B.1.5 Scalability to Big Data

We expressed the above relationships in terms of data size, N, as well as H, since an argument opposing ours could examine $\mathbf{\Sigma}_{prior}^2\mathbf{\Sigma}_{like}^{-1}$ under infinite data. Increasing N shrinks likelihood variance, which could imply that in settings with large data, H would have to be further increased for the inference method to remain valid. This would jeopardise the scalability of anchored ensembles.

Fortunately, as seen in the above analysis, the ratios of interest are N/H^2 and N/H^3, meaning the ratio decreases faster with increasing H than with increasing N, making this a manageable issue. For example if N was increased one hundred-fold, H would only have to be increased by five to ten times. It is anyway normal to increase the size of NNs for larger data sets, so this issue has minimal impact on scalability of the method.

B.2 Joint Parameter Likelihood

The previous section analysed the case of single parameter likelihoods conditional on other parameter values being fixed. This only partially explains why anchored ensembles work. Here we provide the rest of the reason by considering the full covariance matrix. We will see that, even as conditional likelihood variances tend to zero, our method remains valid for large H. This is due to an increasingly high probability of similar parameter pathways through the NN, and hence perfect correlations between parameters likelihoods.

B.2.1 Conditional Likelihoods of Zero Variance

Eq. 43 & 46 suggest that as σ_ϵ^2 reduces to zero, conditional likelihood variance tends to zero, $\frac{\sigma_{prior}^4}{\sigma_w^2} \to \infty$, and hence $\mathbf{\Sigma}_0 \neq \mathbf{\Sigma}_{prior}$. If this is ignored, and we anyway set $\mathbf{\Sigma}_0 = \mathbf{\Sigma}_{prior}$, we might expect to severely underestimate the posterior variance. However, experiments in the main paper, both in section 5.4, and figures 2 & 5, showed this was not the case with σ_ϵ^2 as low as $1e-3$ and $1e-7$.

The reason for this is somewhat surprising. Multicollinearity is a problem in multiple linear regression where input variables are strongly correlated with one another. This makes determining coefficients troublesome, and is realised by a broad, highly correlated, joint likelihood distribution. Coincidentally this is exactly the requirement for $\mathbf{\Sigma}_0 = \mathbf{\Sigma}_{prior}$ to be valid. NNs, having parallel tree-like structures, do have strong correlations amongst parameters; ironically this is the downfall of mean-field VI (section 2.1), but a useful property for anchored ensembles. We illustrate this with the following example.

B.2.2 Joint Likelihood Example

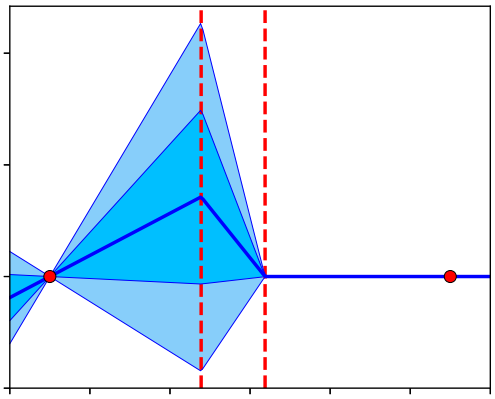


Figure 8: Analytical Predictive Distribution of a Single-Layer NN with Two Hidden Nodes. Dashed Red Lines Show Activation Points for Each Node.

Consider a dataset of two data points, modelled with a single-layer NN of two hidden nodes with ReLU activations, and $\sigma_\epsilon^2 = 1e-9$. Here conditional likelihood variances will be almost zero (since $\sigma_\epsilon^2 \approx 0$), and hence we might expect anchored ensembles to fail. However, if the point where the hidden nodes becomes greater than zero (the activation points), for both nodes, falls in between the two data points, and the active half of the output is also shared, the final layer weights have perfect multicollinearity. We share key info below.

$W_1 = [[-0.813], [-0.396]], b_1 = [-0.993, 0.148], W_2 = [[-0.467], [0.674]], \mathbf{X} = [[-5], [5]], \mathbf{y} = [0, 0], \mathbf{\Sigma}_{prior} = [[0.5, 0.], [0., 0.5]], \mathbf{X}_{in} = max(\mathbf{X}\mathbf{W}_1 + \mathbf{b}_1, 0) = [[[3.070], [2.127]], [[0.], [0.]]]$.

Analytical posterior covariance of final layer weights: $\mathbf{\Sigma}_{post} = (\mathbf{\Sigma}_{prior}^{-1} + \mathbf{X}_{in}\mathbf{X}_{in}^T/\sigma_\epsilon^2)^{-1}$, =

$$\begin{bmatrix} 0.1621968 & -0.23407391 \\ -0.23407391 & 0.33780321 \end{bmatrix}$$

The analytical anchored ensembling posterior covariance when $\mathbf{\Sigma}_0 = \mathbf{\Sigma}_{prior}$ can be found by replacing $\mathbf{\Sigma}_0$ with $\mathbf{\Sigma}_{prior}$ in eq. 38 and simplifying. This results in, $\mathbf{\Sigma}_{post}^{MAP} = \mathbf{\Sigma}_{post}\mathbf{\Sigma}_{post}^T\mathbf{\Sigma}_{prior}^{-1} =$

$$\begin{bmatrix} 0.16219679 & -0.23407391 \\ -0.23407391 & 0.33780321 \end{bmatrix}$$

As we can see, anchored ensembling recreates the true posterior covariance almost perfectly even though the conditional likelihood variance is close to zero. These results have been demonstrated for final layer weights since the likelihood distribution can be found conveniently with $\mathbf{X}\mathbf{X}^T$ as for multiple linear regression. Strong correlations across all parameters means this argument applies equally to parameters in all layers.

The condition for this to occur is that some strong correlations exist between parameters. As H grows, there are more opportunities for similarities in pathways, and hence the probability of high correlations existing increases. This all has the effect of increasing entries in the full covariance matrix, $\mathbf{\Sigma}_{like}$, which suggests $\lim_{H\to\infty} \mathbf{\Sigma}_{prior}^2\mathbf{\Sigma}_{like}^{-1} \to 0$.

C FURTHER RESULTS

Table 2 & 3 shows all variants of our method that were run on the benchmarking datasets. ReLU GP and 5x 50 NNs NLL results are as in table 1. The below discussion focuses on NLL results.

ERF GP refers to the equivalent GP for an infinite width, single-layer BNN with ERF activations. It was tuned and implemented as for the ReLU GP. We were interested to discover how different activation functions would affect uncertainty estimates, since they impose differing assumptions about the data generating function. In general the ReLU GP performed better than the ERF GP, with some exceptions, such as for Wine. The target variable for Wine is ordinal, containing five factors, it is therefore no surprise that the ReLU GP, which extrapolates linearly, is at a slight disadvantage.

10x 50 NNs refers to an anchored ensemble of ten NNs with 50 hidden nodes. We find that these results fall in between the 5x 50 NNs and the ReLU GP. This agrees with the convergence analysis done in sections 5.2 & 5.3.

To prove our method is suitable for more than single-layer NNs, we implemented an anchored ensemble of five double-layer NNs, 5x 50-50 NNs. Even with minimal hyperparameter tuning (section E) we found an extra layer gave a performance boost over the 5x 50 NNs. We expect with more careful tuning this margin would increase.

Single 50 NN refers to a single regularised NN, of one hidden layer with 50 hidden nodes, for which we used a constant value of predictive variance. Although this performs poorly in several cases, e.g. Boston and Yacht, the results are surprisingly close to those achieved by both our method and Deep Ensembles, even surpassing them on the Energy dataset. This sparked our critique of the datasets in section F: a method outputting constant predictive variance should not perform well in experiments designed to test uncertainty quantification.

Table 2: Variants of Our Method on Benchmark Datasets, RMSE.

			RMSE					
	N	D	ReLU GP	ERF GP	5x 50 NNs	10x 50 NNs	5x 50-50 NNs	Single 50 NN
Boston	506	13	2.86 ± 0.16	2.94 ± 0.18	3.09 ± 0.17	3.09 ± 0.17	3.00 ± 0.18	3.40 ± 0.20
Concrete	1,030	8	4.88 ± 0.13	5.21 ± 0.12	4.87 ± 0.11	4.73 ± 0.11	4.75 ± 0.12	5.17 ± 0.13
Energy	768	8	0.60 ± 0.02	0.78 ± 0.03	0.35 ± 0.01	0.34 ± 0.01	0.40 ± 0.01	0.40 ± 0.01
Kin8nm	8,192	8	0.07 ± 0.00	0.08 ± 0.00	0.07 ± 0.00	0.07 ± 0.00	0.06 ± 0.00	0.07 ± 0.00
Naval	11,934	16	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
Power	9,568	4	3.97 ± 0.04	3.94 ± 0.04	4.07 ± 0.04	4.07 ± 0.04	4.03 ± 0.04	4.23 ± 0.04
Protein	45,730	9	4.34 ± 0.02	4.23 ± 0.02	4.36 ± 0.02	4.34 ± 0.02	4.23 ± 0.02	4.56 ± 0.02
Wine	1,599	11	0.61 ± 0.01	0.60 ± 0.01	0.63 ± 0.01	0.62 ± 0.01	0.62 ± 0.01	0.64 ± 0.01
Yacht	308	6	0.60 ± 0.08	1.48 ± 0.15	0.57 ± 0.05	0.54 ± 0.05	0.85 ± 0.08	0.81 ± 0.07
Song Year	515,345	90	9.01 ± NA	8.90 ± NA	8.82 ± NA	8.82 ± NA	8.66 ± NA	8.77 ± NA

Table 3: Variants of Our Method on Benchmark Datasets, NLL.

		NLL					
	$\hat{\sigma}_\epsilon^2$	ReLU GP	ERF GP	5x 50 NNs	10x 50 NNs	5x 50-50 NNs	Single 50 NN
Boston	0.08	2.45 ± 0.05	2.46 ± 0.05	2.52 ± 0.05	2.50 ± 0.05	2.50 ± 0.07	2.70 ± 0.05
Concrete	0.05	2.96 ± 0.02	3.06 ± 0.02	2.97 ± 0.02	2.94 ± 0.02	2.94 ± 0.02	3.08 ± 0.03
Energy	1e-7	0.86 ± 0.02	1.06 ± 0.03	0.96 ± 0.13	0.52 ± 0.06	0.61 ± 0.07	0.57 ± 0.03
Kin8nm	0.02	-1.22 ± 0.01	-1.17 ± 0.00	-1.09 ± 0.01	-1.16 ± 0.01	-1.25 ± 0.01	-1.17 ± 0.01
Naval	1e-7	-10.05 ± 0.02	-9.66 ± 0.04	-7.17 ± 0.03	-7.29 ± 0.02	-7.08 ± 0.13	-6.58 ± 0.04
Power	0.05	2.80 ± 0.01	2.79 ± 0.01	2.83 ± 0.01	2.83 ± 0.01	2.82 ± 0.01	2.86 ± 0.01
Protein	0.5	2.88 ± 0.00	2.86 ± 0.00	2.89 ± 0.01	2.88 ± 0.01	2.86 ± 0.01	2.94 ± 0.00
Wine	0.5	0.92 ± 0.01	0.91 ± 0.01	0.95 ± 0.01	0.94 ± 0.01	0.94 ± 0.01	0.97 ± 0.01
Yacht	1e-7	0.49 ± 0.07	1.50 ± 0.13	0.37 ± 0.08	0.18 ± 0.03	0.04 ± 0.08	1.50 ± 0.02
Song Year	0.7	3.62 ± NA	3.61 ± NA	3.60 ± NA	3.60 ± NA	3.57 ± NA	3.59 ± NA

D ADDITIONAL PLOTS

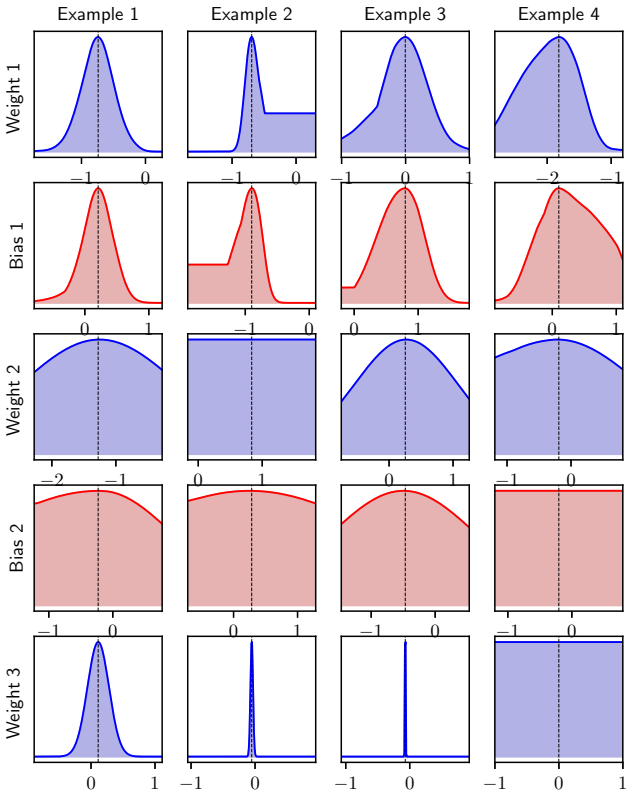


Figure 9: Likelihood Plot for a Two-Hidden Layer NN on Toy Dataset, ReLU Activations. Note the Piece-wise Likelihood Shape Caused by Data Points Crossing ReLU Elbows as in Example 2, Weight 1.

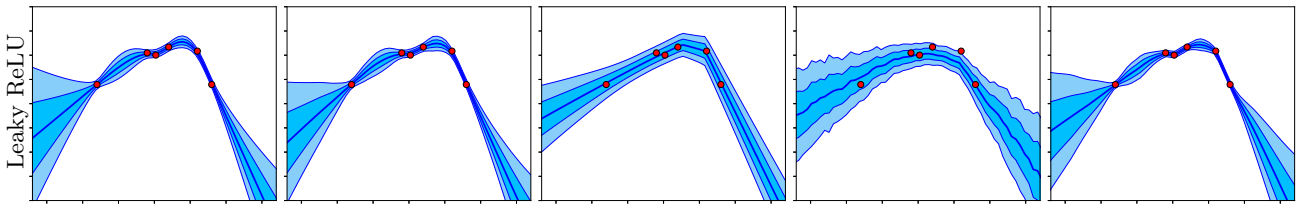


Figure 10: Predictive Distributions Produced by Various Inference Methods (Columns) with Leaky ReLU Activations in Wide Single-Layer NNs: Data and Methods As for Figure 2.

E EXPERIMENTAL DETAILS

E.1 Introduction to Anchored Ensembles

Experimental details for figure 1 are as follows.

Six randomly generated data points were used.

Hyperparameters: activation = ERF, $\sigma_\epsilon^2 = 0.003$, b_1 variance = 1, W_1 variance = 1, $H = 100$, $M = 3$ (number of ensembles), optimiser = adam, epochs = 400, learning rate = 0.005.

E.2 Panel of Inference Methods

Experimental details for figure 2 are as follows.

Same six data points were used for all methods and activation functions, generated by $y = x\sin(5x)$, evaluated at, [-0.8, -0.1, 0.02, 0.2, 0.6, 0.8].

Hyperparameters: b_1 variance = 10, W_1 variance = 10, $H = 100$, $M = 10$, epochs= 4,000, $\sigma_\epsilon^2 = 0.001$, leaky ReLU $\alpha = 0.2$, optimiser = adam, MC Dropout probability = 0.4, MC Dropout samples = 200, HMC step size = 0.001, HMC no. steps = 150, HMC burn in = 500, HMC total samples = 1000, HMC predict samples = 50, VI predict samples = 50, VI iterations = 2000, VI gradient samples = 200.

The RBF case is of slightly different form than that given in eq. 3, we adopt the notation used in Williams (1996), with U variance = 2, g variance = 0.5 (and untrainable).

The predicted epistemic uncertainty is taken from models, and we subsequently add on data noise (this was done for all experiments).

E.3 Regularisation Dilemma

Experimental details for figure 3 are as follows.

Generated $\mathbf{X}$ by sampling 20 points linearly spaced from the interval [-1.5, 1.5], $y = sin(2x) + \epsilon$ with $\epsilon \sim \mathcal{N}(0, 0.2^2)$. The y value corresponding to the largest x value was shifted -0.4 to produce a slight outlier. Sub-plot A was trained via mean square error, B was regularised, C was anchored.

Hyperparameters: activation = ReLU, $\sigma_\epsilon^2 = 0.08$, b_1 variance = 10, W_1 variance = 10, $H = 1000$, optimiser = adam, epochs = 2,000, learning rate = 0.003, $M = 5$.

E.4 1-D Convergence Plots

Experimental details for figure 5 are as follows.

Data as in section E.2 was used, with M = [3,5,10,20].

Hyperparameters: activation = ReLU, $\sigma_\epsilon^2 = 0.001$, b_1 variance = 20, W_1 variance = 20, $H = 100$, optimiser = adam, epochs = 4,000, learning rate = 0.005.

E.5 KL Convergence Results

Experimental details for figure 6 are as follows.

This was on Boston Housing dataset, where 50% was used for training, with testing on other 50%.

Hyperparameters: activation = ReLU, $\sigma_\epsilon^2 = 0.1$, b_1 variance = 2, W_1 variance = 2, H = [4, 16, 64, 256, 1024], M = [3,5,10,20,40], optimiser = adam, no. runs = 10, epochs = 1,000, learning rate = 0.001 when $H < 20$ else learning rate = 0.0002.

E.6 Likelihood Plots

Experimental details for figure 7 are as follows.

Table 4: Hyperparameters Used For Regression Benchmark Results.

	N	Batch Size	Learn Rate	$\hat{\sigma}_\epsilon^2$	b_1 variance	W_1 variance	No. Epochs	Decay Rate	Single NN var.
Boston	506	64	0.05	0.06	10	0.77	3000	0.995	0.45
Concrete	1,030	64	0.05	0.05	40	5.00	2000	0.997	0.28
Energy	768	64	0.05	1e-7	12	1.50	2000	0.997	0.03
Kin8nm	8,192	256	0.10	0.02	40	5.00	2000	0.998	0.32
Naval	11,934	256	0.10	1e-7	200	12.50	1000	0.997	0.03
Power	9,568	256	0.20	0.05	4	1.00	1000	0.995	0.24
Protein	45,730	8192	0.10	0.5	50	5.56	3000	0.995	0.71
Wine	1,599	64	0.05	0.5	20	1.82	500	0.997	0.77
Yacht	308	64	0.05	1e-7	15	2.50	3000	0.997	0.10
Song Year	515,345	32768	0.01	0.7	2	0.02	500	0.996	0.84

We trained and tested a two-layer NN over the full Boston dataset, minimising mean squared error (unconstrained).

Hyperparameters: activation = ReLU, $\sigma_\epsilon^2 = 0.1$, b_1 variance = 0.08, W_1 variance = 0.08, optimiser = adam, epochs = 9,000, learning rate = 0.0003.

H was varied from [5, 50, 500]. For each value of H we randomly selected two of the parameters for each layer and type. For each of these, we fixed all other parameters in the NN, and varied the selected parameter by it's learnt value $\pm$ 0.3, calculating the total likelihood of the data at each parameter setting. In this way the likelihoods could be plotted.

E.7 Main Benchmarking Experiments

The hyperparameter tuning process and final settings for experiments in table 1, 2 & 3 are given as follows.

E.7.1 Hyperparameter Tuning

Hyperparameter tuning was done on a single train/validation split of 80%/20%. We found it convenient to begin by tuning data noise variance and prior variances. We restricted the prior variance search space by enforcing W_1 variance = b_1 variance / D, and W_2 variance = $1/H$. We therefore had only two hyperparameters to optimise initially: b_1 variance and σ_ϵ^2. We did this with the GP model, using grid search, maximising marginal log likelihood over the training portion, and minimising NLL of the validation portion. For the larger datasets, when inference over the 80% training portion was too slow, we reduced the training split to 2,000 data points.

Hyperparameters for priors and data noise estimates were shared between the GP and anchored ensembles. Hyperparameters requiring tuning specifically for anchored ensembles were batch size, learning rate, number of epochs and decay rate. This was done on the same 80%/20% split used to select data noise and prior variance. We used random search, directed by our knowledge of the optimisation process (e.g. a lower learning rate requires more epochs to converge), minimising NLL on the validation portion.

We did not retune hyperparameters from scratch for the double layer NN (5x 50-50 NNs). We used settings as for the single-layer NNs (5x 50 NNs), but divided learning rate by 4, and multiplied epochs by 1.5.

For the single regularised NN with constant noise, we again used hyperparameters as for the single-layer ensemble (5x 50 NNs), tuning only the constant amount of variance to be added on the same 80%/20% split.

E.7.2 Hyperparameter Settings

Table 4 provides the key hyperparameters used. The adam optimiser was used for all experiments. ReLU activations were used for all except the ERF GP (prior variance was separately tuned for this, values aren't given in the table).

F DATASET CRITIQUE

In section 5.4 we referred to weaknesses in the benchmarking experiments. These rose to our attention following results in section C showing that a method outputting constant variance for every prediction performed well on some datasets. We believe this should not be the case for a good test of uncertainty quality.

Our main criticism is that the benchmarking experiments *only* test the ability of a model to predict a well calibrated distribution for data drawn from the same distribution seen during training. An arguably more important ability is to output high uncertainty when a new instance, very different to that seen in training, occurs. This second ability is what drives the oft-cited applications of outlier detection, exploration in RL, and active learning.

Some benchmarking datasets do test this to some extent whilst others don't at all. To illustrate this, we plot a scatter matrix of four of them in figure 11. The Energy and Yacht datasets appear to have used grid-like input settings to create each data point. This means that whilst they contain a reasonable number of variables,

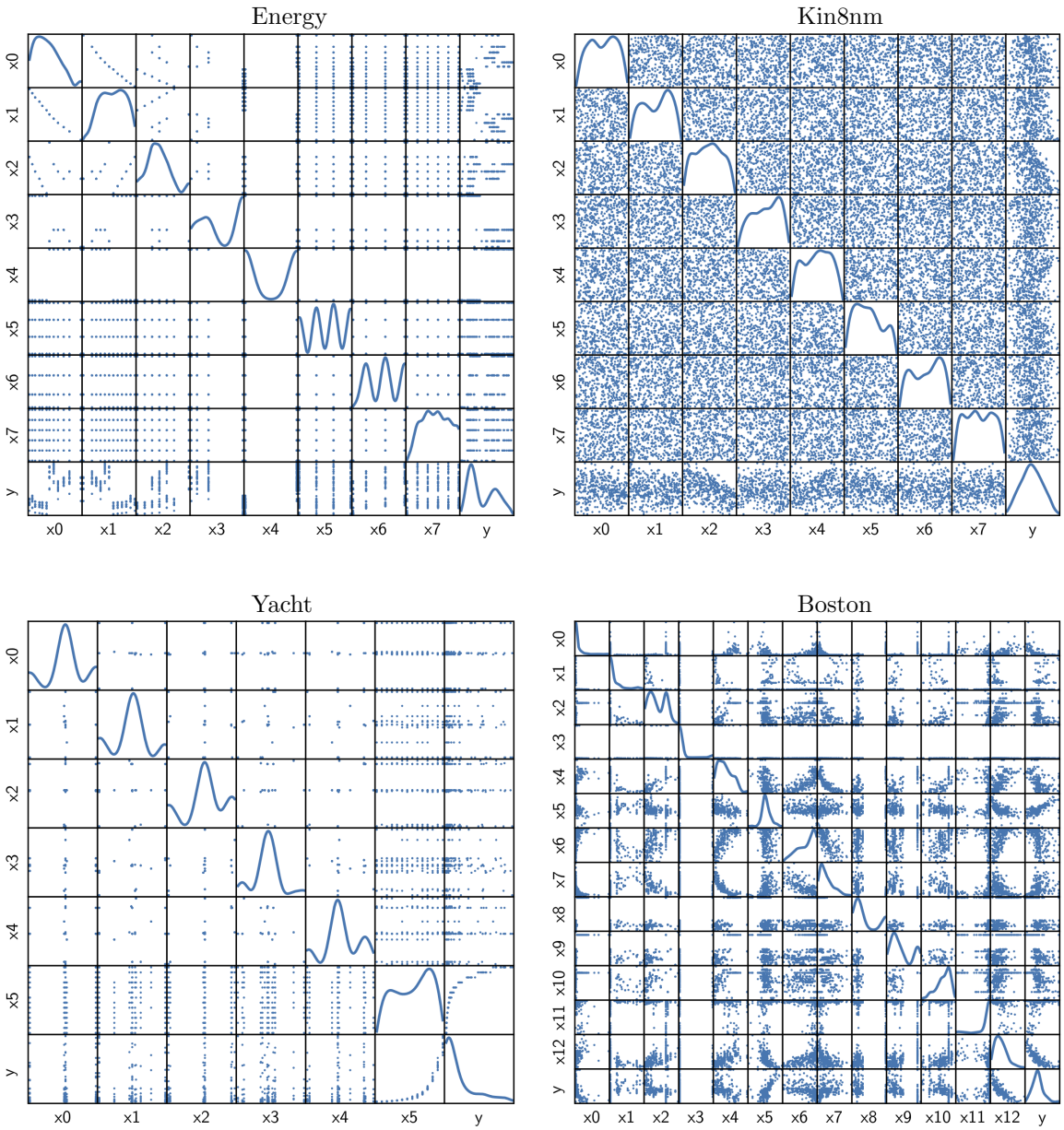


Figure 11: Matrix of Scatter Plots for Four Benchmark Datasets.

the effective manifold area is very small and neither extrapolation nor interpolation are tested to any great degree. On the other hand, the Kin8nm dataset appears to have independently sampled each input feature from a uniform distribution, a manifold unlikely to occur in the real world. This also results in no opportunity for extrapolation to be tested.

These weaknesses may arise from the fact that both Energy and Kin8nm were simulated datasets. We have also plotted Boston, the classic house pricing dataset, as an example of a more useful dataset - we see a complex, noisy manifold, with some relative outliers in the data, which test extrapolation of uncertainty.

Note the analysis in this section was done after completion of our experiments.