Approximately Bayesian Ensembling

Amritha Sukhdev Singh Agarwal
Technische Universität Dortmund
Dortmund, North-Rhine Westphalia, Germany

ABSTRACT

There are several applications where understanding the uncertainty of a neural network's (NN) predictions is critical, and measuring this in NNs is a difficult yet unsolved challenge. There are various existing methods for dealing with NN uncertainty. The Approximately Bayesian Ensembling is one such strategy. The paper presented a modification to the standard ensembling methodology that enables the ensemble to execute Bayesian inference, leading to convergence to the appropriate Gaussian Process when the overall number of NNs and their sizes tend to infinity. In a simplified situation, the recovered posterior is correctly centered, but marginal variance tends to have underestimated marginal variance and overestimated correlation. Two circumstances, though, may result in precise recovery. The paper illustrates out-of-distribution data with classification through experiments and shows that the method is competitive with variational approaches and has an advantage over traditional ensembling. A theoretical analysis of the approach in a simplified situation indicates that the recovered posterior is correctly centered but tends to have underestimated marginal variance and overestimated correlation. Two circumstances, though, may result in precise recovery. The paper illustrates out-of-distribution data with classification through experiments and shows that the method is competitive with variational approaches and has an advantage over traditional ensembling.

KEYWORDS

Uncertainty, Neural Networks, Bayesian NNs, Ensembling

ACM Reference Format:
Amritha Sukhdev Singh Agarwal. 2018. Approximately Bayesian Ensembling. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation emai (Conference acronym 'XX)*. ACM, New York, NY, USA, 7 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

With state-of-the-art results across domains, neural networks are fast becoming the dominant approach in machine learning. However, the standard neural networks do not incorporate a mechanism for determining individual predictions' certainty (or uncertainty) since they are not probabilistic. This is significant as uncertainty quantification is essential for many real-world applications.

The most widely used approach to dealing with uncertainty is the Bayesian framework provided by Ghahramani [3], wherein the weights are modeled as distributions and determined using the Bayes rule. Although BNNs are a viable solution, modern NNs typically have millions of parameters and data points, making them costly and inefficient.

Ensembling is a non-bayesian method for handling uncertainty that involves training a small ensemble of NNs, starting from various initializations and occasionally using noisy training data. When given new data to predict, there will be some variation in the ensemble's predictions, which can be seen as uncertain. The reasoning behind this is straightforward: if the new data point is comparable to the training data, all of the NNs should produce estimates similar to each other, but if it is significantly different, there should be a more considerable variance in the predictions.

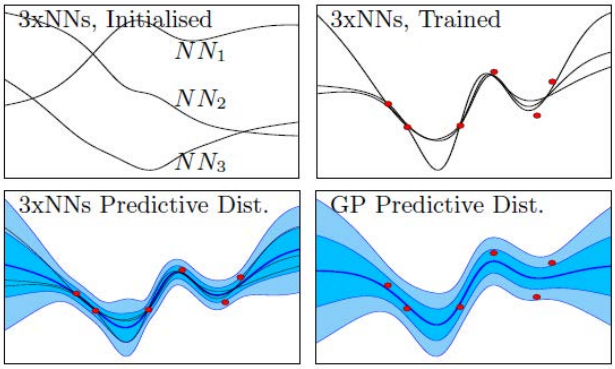

Figure 1: An ensemble of NNs starting from various initializations points.

Even though ensembling has provided good empirical results [4] and is simple to scale and apply, the departure from the fundamental Bayesian methodology is alarming.

The paper tries to combine the paradigms of Ensembling with Bayesian Neural Networks by modifying the usual NN ensembling procedure to align with Bayesian inference. The proposed modification alters the loss function and adds $\theta_{anc,j}$ randomization, whose value is equal to the prior distribution, in place of the L2 regularization. This approach, known as anchored ensembling, belongs to the family of methods known as randomized MAP sampling (RMS).

The methodology employed in the paper, the Randomized MAP sampling, is described in the next section before its theoretical analysis. The application of RMS to NNs—also known as anchored ensembling—with a practical workaround is covered in Sections 3 and 4. The experiment results are covered in the fifth part. The sixth section concludes the research, examines potential additional analysis, and summarizes the key findings

Conference acronym 'XX, June 03–05, 2018, Woodstock, NY

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

2 METHODOLOGY

2.1 Bayesian Neural Network

Specifying an acceptable model is a crucial step in the Bayes' rule framework for handling uncertainty. In NNs, however, there is challenging to convert prior beliefs about a task into priors over parameters. This paper contributes to previous work that discussed the behavior of Gaussian Processes (GPs) and Bayesian Neural Networks. For example, Radford Neal [6] noted that BNNs with unlimited width would eventually converge to GPs. This relationship is crucial to this study because it makes it possible to examine how a BNNs parameter distribution translates into distribution across functions. This is useful since it offers a more comprehensible space for selecting priors for a BNN.

2.2 Randomized MAP Sampling

A novel approach to performing Bayesian inference is the Randomized MAP sampling method. It refers to the fact that a regularization term added to the loss function yields a Maximum a posteriori (MAP) estimate, which is nothing more than a point estimate of the posterior. The key idea is to sample several times while adding the noise. This noise can either be to the targets or the regularization term. As a result, the distribution of estimates that closely resembles the true posterior is produced.

Although this approach is simple, applying it to NNs or classification settings seems challenging. Despite not reflecting the true posterior, this method has had some empirical success from the linear case to NNs but wrapping the optimization step in an MCMC technique offers a more accurate and computationally costly solution. This group of schemes is known as randomized MAP sampling.

3 RMS THEORETICAL RESULTS

This section presents the original findings of the paper. First, given the simplification that both the prior and the parameter likelihood are multivariate normal distributions, we develop a general version of RMS by analyzing the method in parameter space.

The RMS can be built to retrieve the true posterior if the parameter likelihood covariance is known beforehand. However, in most cases, this will not be known; hence a helpful solution that requires knowledge of the prior distribution is suggested.

This practical workaround prevents returning the true posterior even in the case of normally distributed data. As a result, the outcome is biased due to an overestimated correlation coefficient, and an underestimated marginal variance. Nevertheless, under two unique circumstances, a precise recovery can occur.

3.1 Parameter-Space derivation

Consider that the prior, $\theta \sim \mathcal{N}(\mu_{prior}, \Sigma_{prior})$, and likelihood $P_\theta(\mathcal{D}|\theta) \propto \mathcal{N}(\theta; \mu_{like}, \Sigma_{like})$ follows a multivariate normal distribution. Two types of likelihood are considered, data likelihood and parameter likelihood, with a distinction between the two. Although they are interchangeable and produce the same values when given a data set $\mathcal{D}$ and parameter values θ, their forms are slightly different. Data likelihood is the term used to describe the likelihood of the data given the parameters. The parameter likelihood is defined as the likelihood of the parameters in the parameter space.

The posterior, which is also normal, is given by the Bayes rule as

$$\mathcal{N}(\mu_{post}, \Sigma_{post}) \propto \mathcal{N}(\mu_{prior}, \Sigma_{prior})\mathcal{N}(\mu_{like}, \Sigma_{like}).$$

The MAP estimate is the mean and is provided in closed form as $\theta_{MAP} = \mu_{post}$ and $\Sigma_{post} = (\Sigma_{like}^{-1} + \Sigma_{prior}^{-1})^{-1}$. The resulting equation is given by

$$\theta_{MAP} = \Sigma_{post}\Sigma_{like}^{-1}\mu_{like} + \Sigma_{post}\Sigma_{prior}^{-1}\mu_{prior}.$$

Noise must be added to the calculation above per RMS. Since the modeler has complete control over this number, the paper suggests employing the prior mean as a source of noise injection. By setting $\mu_{prior} = \theta_{anc}$, the above equation yields

$$f_{MAP}(\theta_{anc}) = \Sigma_{post}\Sigma_{like}^{-1}\mu_{like} + \Sigma_{post}\Sigma_{prior}^{-1}\theta_{anc}.$$

Here, f_{MAP} becomes a function that takes in the random variable θ_{anc} and returns a MAP estimate. The tricky part is picking a distribution for the anchor distribution θ_{anc} so that the distribution of functions closely resembles our real posterior. This can be done by setting $\mathbb{E}\left[f_{\text{MAP}}\left(\theta_{\text{anc}}\right)\right] = \mu_{\text{post}}$ and $\text{Var}\left[f_{\text{MAP}}\left(\theta_{\text{anc}}\right)\right] = \Sigma_{\text{post}}$ in the following theorem in order to get μ_{anc} and Σ_{anc}:

Theorem 1. If $P\left(f_{MAP}\left(\theta_{\text{anc}}\right)\right) = P(\theta \mid \mathcal{D})$, then θ_{anc} also follows a multivariate normal distribution with $P\left(\theta_{\text{anc}}\right) = \mathcal{N}\left(\mu_{anc}, \Sigma_{\text{anc}}\right)$, where

$$\mu_{\text{anc}} = \mu_{\text{prior}}$$

$$\Sigma_{\text{anc}} = \Sigma_{\text{prior}} + \Sigma_{\text{prior}}\,\Sigma_{\text{like}}^{-1}\,\Sigma_{\text{prior}}\,.$$

Figure 2 provides the 2D parameter space demonstration for the RMS algorithm.

3.2 Practical Workaround: General Case

The method for creating a Randomized MAP sampling algorithm that will exactly recover the real Bayesian posterior was demonstrated in the preceding section. Unfortunately, the likelihood covariance parameter must be known to set the anchor distribution, which is impossible for most models.

Theorem 1's second term can be ignored as a workaround by setting $\Sigma_{anc} := \Sigma_{prior}$. Although using RMS with this anchor distribution will not usually result in an accurate recovery of the true posterior, the resulting distribution may be considered an approximation, which can be derived in Corollary 1.1.

Corollary 1.1 The RMS approximate posterior is given as - $P(f_{MAP}(\theta_{anc})) = \mathcal{N}(\mu_{post}, \Sigma_{post}\Sigma_{prior}^{-1}\Sigma_{post})$ by setting $\mu_{anc} = \mu_{prior}$ and $\Sigma_{anc} = \Sigma_{prior}$. The proof uses a methodology similar to Theorem 1, except we set $\mu_{anc} = \mu_{prior}$ and $\Sigma_{anc} = \Sigma_{prior}$, and solve for $\mathbb{E}[f_{\text{MAP}}(\theta_{\text{anc}})]$ and $\text{Var}[f_{\text{MAP}}(\theta_{\text{anc}})]$ instead of setting and solving for μ_{anc} and Σ_{anc}.

Despite the covariances of the two distributions not matching, this corollary demonstrates that their means do.

3.3 Practical Workaround: Special Cases

After outlining the covariance bias that typically exists in the RMS approximation posterior, this section provides two unique circumstances in which there is no bias and the genuine posterior is precisely retrieved, as demonstrated in figure 3 (B, C).

The two special requirements are accurate recovery using extrapolation parameters and perfect correlations. Extrapolation parameters are model parameters that do not affect a training dataset's

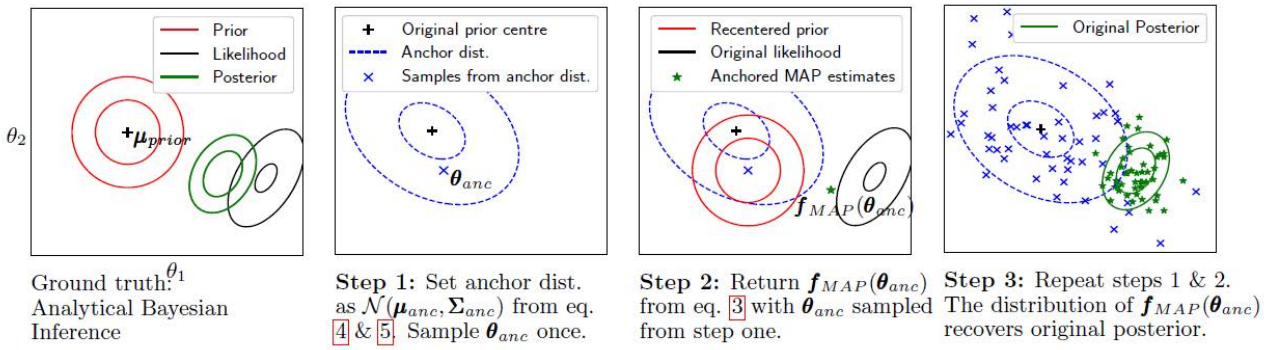

Figure 2: (Exact) RMS in a 2D parameter space.

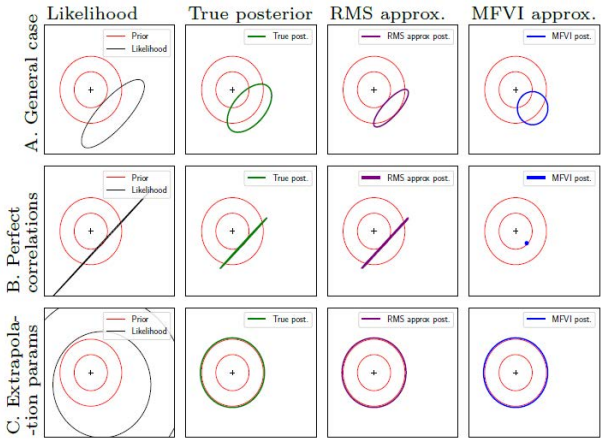

Figure 3: Examples of the RMS approximate posterior along with MFVI.

data likelihood but may impact model predictions on a new data point.

Theorem 3 and 4 in the paper provide proof for the extrapolation parameters by deriving that the marginal RMS approximation posterior equals the marginal true posterior when the extrapolation parameters of a model are set to $\mu_{anc} := \mu_{prior}$, $\Sigma_{anc} := \Sigma_{prior}$.

The proofs in this section demonstrate that RMS performs an accurate recovery if these two requirements hold. As these conditions get closer, one would anticipate seeing an RMS approximation that is more accurate.

4 RMS FOR NEURAL NETWORKS

This section focuses on using the RMS's practical workaround for NNs, known as anchored ensembling [7]. First, the RMS-corresponding NN loss function that needs to be optimized is defined. Following that, given the presumptions, the validity of the RMS process in the context of NNs is examined. Finally, issues arising throughout the scheme's implementation are considered. The corresponding algorithm is provided in figure 4.

Algorithm 1 Implementing anchored ensembles of NNs

Input: Training data, $\mathbf{X}$ & $\mathbf{Y}$, test data point, $\mathbf{x}^*$, prior mean and covariance, $\boldsymbol{\mu}_{\text{prior}}$, $\boldsymbol{\Sigma}_{\text{prior}}$, ensemble size, M, data noise variance estimate, $\hat{\sigma}^2_\epsilon$ (regression only).
Output: Estimate of mean and variance, $\hat{\mathbf{y}}$, $\hat{\sigma}^2_y$ for regression, or class probabilities, $\hat{\mathbf{y}}$ for classification.

 # Set regularisation matrix
$\boldsymbol{\Gamma} \Leftarrow \hat{\sigma}^2_\epsilon \boldsymbol{\Sigma}^{-1}_{\text{prior}}$ (regression) OR $\boldsymbol{\Gamma} \Leftarrow \frac{1}{2}\boldsymbol{\Sigma}^{-1}_{\text{prior}}$ (classification)
 # Create ensemble
$\boldsymbol{\mu}_{\text{anc}} \Leftarrow \boldsymbol{\mu}_{\text{prior}}, \boldsymbol{\Sigma}_{\text{anc}} \Leftarrow \boldsymbol{\Sigma}_{\text{prior}}$
for $j = 1$ **to** M
  $\boldsymbol{\theta}_{anc,j} \sim \mathcal{N}(\boldsymbol{\mu}_{\text{anc}}, \boldsymbol{\Sigma}_{\text{anc}})$ *# Sample anchor points*
  NN_j.create($\boldsymbol{\Gamma}, \boldsymbol{\theta}_{anc,j}$) *# Create custom regulariser*
  NN_j.initialise() *# Initialisations independent of* $\boldsymbol{\theta}_{anc,j}$
 # Train ensemble
for $j = 1$ **to** M
  NN_j.train($\mathbf{X}, \mathbf{Y}$), loss in eq. 4.9 (regression) or eq. 4.10 (classification) or eq. 4.7 (custom)
 # Predict with ensemble
for $j = 1$ **to** M
  $\hat{\mathbf{y}}_j \Leftarrow NN_j$.predict($\mathbf{x}^*$)
 # Regression - combine ensemble estimates
$\hat{\mathbf{y}} = \frac{1}{M}\sum_{j=1}^{M} \hat{\mathbf{y}}_j$, *# 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. = epistemic + data noise*
 # Classification - combine ensemble estimates
$\hat{\mathbf{y}} = \frac{1}{M}\sum_{j=1}^{M} \hat{\mathbf{y}}_j$, *# Average softmax output*
$\hat{\sigma}^2_y$ = None *# N/A for classification*
return $\hat{\mathbf{y}}, \hat{\sigma}^2_y$

Figure 4: Algorithm for Anchored Ensembles.

4.1 Loss Function

A NN with parameters, θ, making predictions($\hat{y}$) with data points, N, and H hidden nodes are considered. Then, the prior distribution is $P(\theta) = \mathcal{N}(\mu_{prior}, \Sigma_{prior})$. Here, we assume that the prior and likelihood follow a normal distribution. The MAP solution is as follows:

$$\theta_{MAP} = \text{argmax}_\theta \, P(\theta \mid \mathcal{D})$$
$$\theta_{MAP} = \text{argmax}_\theta \, P_{\mathcal{D}}(\mathcal{D} \mid \theta) P(\theta)$$
$$\theta_{MAP} = \text{argmax}_\theta \log\left(P_{\mathcal{D}}(\mathcal{D} \mid \theta)\right) + \log(P(\theta))$$

$$\theta_{MAP} = \text{argmax}_\theta \log\left(P_{\mathcal{D}}(\mathcal{D} \mid \theta)\right) - \frac{1}{2}\left\|\Sigma^{-1/2}_{prior}(\theta - \mu_{prior})\right\|^2_2$$

Standard L2 regularization occurs when $\mu_{prior} = 0$, but to apply RMS, substitute μ_{prior} with θ_{anc}

The MAP estimate for the Regression task is given as follows:

$$\text{Loss}_j = \frac{1}{N} \left\| \mathrm{y} - \hat{\mathrm{y}}_j \right\|_2^2 + \frac{1}{N} \left\| \Gamma^{1/2} \cdot (\theta_j - \theta_{anc,j}) \right\|_2^2 .$$

Here, homoskedastic Gaussian noise of variance σ_ϵ^2 is assumed for the task. The diagonal regularisation matrix, Γ is $\text{diag}(\Gamma)_i = \sigma_\epsilon^2 / \sigma^2_{\text{prior}\ i}$, and j stands for an ensemble of M NNs $j \in \{1 \dots M\}$ each with a unique draw of θ_{anc}.

Cross entropy is typically maximized for classification applications, assuming a multinomial data likelihood, and is given as

$$\text{Loss}_j = -\frac{1}{N} \sum_{n=1}^{N} \sum_{c=1}^{C} y_{n,c} \log \hat{y}_{n,c,j} + \frac{1}{N} \left\| \Gamma^{1/2} \cdot (\theta_j - \theta_{anc,j}) \right\|_2^2 ,$$

. Here, the class label y_c for $c \in \{1 \dots C\}$ classes and Here, $\text{diag}(\Gamma)_i = 1/2\sigma^2_{\text{prior}\ i}$.

4.2 Validity of RMS in NNs

The multivariate normal distribution assumptions are validated in this section, along with the special conditions that lead to close approximations of the actual posterior distributions.

4.2.1 Normal distribution. It was previously assumed that parameter likelihoods follow a multivariate normal distribution. Two reasons are provided for using this assumption in NNs.

1) Other approximate Bayesian methods like MFVI and the Laplace approximation [8] make similar assumptions. It is typical for MFVI to fit a factorized normal distribution to the posterior. The Laplace distribution fits the mode of MAP solutions to a multivariate normal distribution.

2) Figure 5 depicts the conditional parameter likelihood for NNs trained on regression and classification. Following training, a random parameter is chosen, and all others are frozen. The choice parameter is adjusted over a narrow range, and the data likelihood is determined at each point. As a result, the conditional distributions are shown. Analyzing local modes as approximately normally distributed seems reasonable based on the plots.

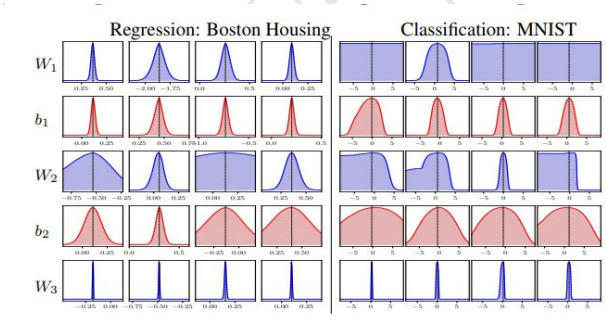

Figure 5: Conditional likelihood graphs for four randomly selected parameters in two-layer NNs.

It is, therefore, justified to model the parameter likelihood as a multivariate normal distribution with a single mode. In the parameter space of a NN, however, such modes are likely to be numerous, with each member of an anchored ensemble ending up at a different mode. Moreover, many of these modes arise from parameter symmetries and would be exchangeable, making the MAP solutions exchangeable.

4.2.2 Special cases. RMS approximate posteriors produced by setting the anchor distribution equal to the prior have, in general, underestimated variance and overestimated correlation. Figure 7 display bias-free predictive distributions for anchored ensembles resembling the real Bayesian predictive posterior.

The two exceptional conditions resulting in exact recovery make the distribution bias-free. First, it should be simple to notice that the figures contain extrapolation parameters. This is because all the concealed nodes in the data range will die. The data likelihood is then unaffected by their corresponding final layer weight; however, they do have an impact on forecasts outside of the training set.

Perfect correlations are more challenging to understand, and a numerical example is demonstrated in the paper. A hidden node must become live between two data points for this to work. The final layer weights linked to them are then perfectly correlated. Later tests with CNNs will indirectly examine if these unique conditions continue beyond fully-connected NNs. Increasing the width of the NN, which adds more parameters and increases the likelihood of significant correlations, is an apparent method to support these requirements further.

4.3 Implementation Practicalities

How many NNs should an RMS ensemble contain? Unfortunately, many samples can only fully capture the posterior parameter distributions. On the other hand, if one considers each NN as an iid sample from a posterior predictive distribution, a significantly fewer number are necessary given that output dimensionality is often low. Notably, the input dimension has no bearing on this. Additionally, the studies employed 5–10 ensembles scaled by $O(MN)$.

Is it necessary to initialize the NNs at anchor points? Although it is practical to derive parameter initializations from the anchor distribution and regularize immediately around these initialized values, the authors discovered that trials were improved when initializations were decoupled from the anchor points.

5 EXPERIMENTS

This section presents vital discoveries from the research.

5.1 Qualitative Tests

The authors initially investigated anchored ensembles on toy problems to get a feel for their behavior compared to standard approximate inference and ensembling techniques.

A comparison of popular Bayesian inference methods within single-layer NNs for ReLU and sigmoidal nonlinearities is presented in figure 6. Bayesian inference produced by GP and HMC is deemed the best by comparison, and all the other methods are judged on how close they are to the gold standard. Interpolated uncertainty is poorly captured by MC dropout and MFVI (with a factorized normal distribution). This is a sign that the posterior approximation ignores parameter correlations [1].

Figure 7 depicts a group of 10NN who were trained on straightforward regression problems using common loss functions, either

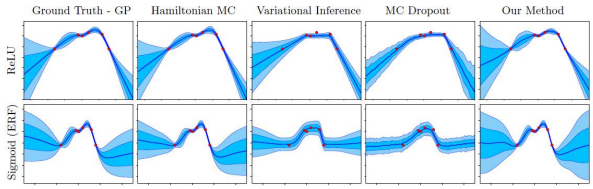

Figure 6: Predictive distributions from single-layer NNs trained on a simple regression problem using a variety of inference techniques.

with no regularization term ('unconstrained', $\Gamma = 0$) or with regularization centered on zero ('regularized', $\theta_{anc,j} = 0$). Since regularization eliminates the ensemble's variety and pushes all NNs toward the same single solution, it gives subpar results. Unconstrained is also improper because, despite producing diversity, it maintains no knowledge of a prior and overfits the data. Unconstrained is most similar to deep ensembles.

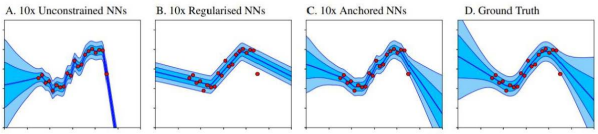

Figure 7: NN ensemble loss comparison for the toy regression challenge.

5.2 Convergence Behaviour

The Boston housing dataset was used to assess how accurately anchored ensembling performs Bayesian inference on real datasets compared with an exact method (ReLU GP). The ensemble's predicted and GP's predictive distribution were compared using the KL divergence method; zero indicates identical distributions. Both models used half of the data for training and a half for testing. Results were averaged over ten runs, with each run's test/train split being shuffled at random. For both models, the data noise variance was fixed at $\sigma_\epsilon^2 = 0.1$. Preprocessing of the data followed the UCI regression technique.

Figure 8 quantifies the change that results from adjusting the ensemble's NNs' width and number. Instead of using anchored NNs, the "ideal" line displays the metric where posterior samples from the GP itself were used. The KL divergence between the two prediction distributions reduces with increasing NN width and NN count. However, a modest residual difference persists even with 40xNNs with 1,024 nodes.

5.3 UCI Regression Benchmarks

A standard BNN benchmark was utilized to compare anchored ensembles to well-known approximate inference techniques. The benchmark aims to minimize negative log-likelihood (NLL) and root means square error (RMSE) for each dataset. Ninety percent of the data is used to train the models, and the remaining 10 percent is used to record RMSE and NLL. Using single-layer NNs with 50 hidden

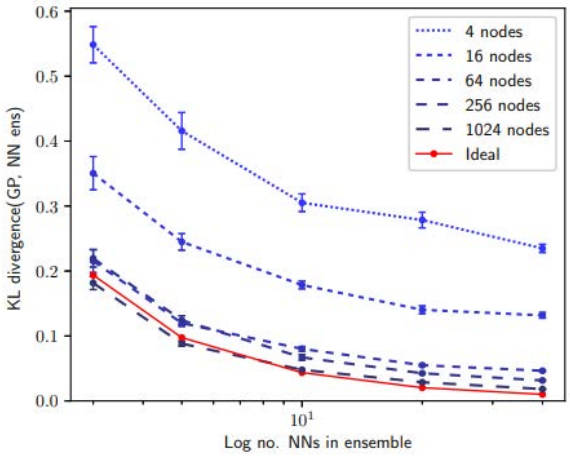

Figure 8: anchored ensemble and a ReLU GP. Mean ±1 standard error.

nodes, experiments are run 20 times with random test/train splits. Only five and one repetitions are allowed for the more significant datasets Protein and Song, which allow for 100 hidden nodes.

Figure 9 displays the outcomes. Additionally, the outcomes for Deep Ensembles [4] were also included. An obvious pattern emerges when ranking results by estimated data noise level σ_ϵ^2 anchored ensembles outperform deep ensembles and deep learning ensembles in datasets with low data noise.

	$\hat{\sigma}_\epsilon^2$	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 comparison 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.

Figure 9: Benchmark results for NLL regression. Mean ±1 standard error.

5.4 Out-of-Distribution Classification

The research on classification tasks for out-of-distribution (OOD) data using complicated NN architectures are presented in this part, along with a comparison to alternative ensemble methods.

Three separate datasets are trained using an NN architecture suitable for each dataset: 1) Classification of fashion images using

three fully connected layers with 100 hidden nodes. 2) Classification of the emotion of IMDb movie reviews using embedding (20 dimensions) and 1D convolution (50 filters, kernel size of 3), followed by a fully-connected layer (200 hidden nodes). 3) Convolutional NN (CNN) with 9 million parameters, 64-64-max pool-128-128-max pool-256-256-256-max pool-512-512-512-max pool-flatten-2048-softmax, CIFAR-10 picture classification. Experiments were run five times for Fashion MNIST and IMDb; for CIFAR-10, they were run three times.

Due to the lack of data augmentation and batch normalization, the accuracy levels remain below the state-of-the-art.

OOD inputs presented to the NNs are shown in figure 10. Two groups held out during training are called Edge (for CIFAR ships, dogs; for Fashion MNIST trousers, sneakers). Each row of pixels in a particular image gets scrambled. Invert used the pixel values that were in the negative, and Noise selected pixels from a large-magnitude Bernoulli distribution (p=0.005) with a pixel value of 50. It was trained on 40K instances.

CIFAR-10 Image Classification, VGG-13 CNN

Train — Edge — Fashion MNIST Scramble Invert Noise

	Accuracy	Train	Edge	Fashion	MNIST	Scramble	Invert	Noise
1xNNs Reg.	81.6%	0.671	0.466	0.440	0.540	0.459	0.324	0.948
5xNNs Uncons.	85.0%	0.607	0.330	0.208	0.275	0.175	0.209	0.380
5xNNs Reg.	86.1%	0.594	0.296	0.219	0.188	**0.106**	0.153	0.598
5xNNs Anch.	85.6%	0.567	**0.258**	**0.184**	**0.149**	0.134	**0.136**	**0.118**
10xNNs Anch.	86.0%	0.549	0.256	0.119	0.145	0.122	0.124	0.161

IMDb Text Sentiment Classification, Embedding+CNN

	Accuracy	Train	Reuters	Rand. 1	Rand. 2	Rand. 3
1xNNs Reg.	85.3%	0.637	0.119	0.153	0.211	0.326
5xNNs Uncons.	89.1%	0.670	0.102	0.141	0.100	0.075
5xNNs Reg.	87.1%	0.612	0.051	0.091	0.076	0.055
5xNNs Anch.	87.7%	0.603	**0.049**	**0.075**	**0.061**	**0.009**

Fashion MNIST Image Classification, Fully-Connected NN

	Accuracy	Train	Edge	CIFAR	MNIST	Distort	Noise
1xNN Reg.	86.8 %	0.660	0.584	0.143	0.160	0.429	0.364
5xNNs Uncons.	89.0 %	0.733	0.581	0.301	0.104	0.364	0.045
5xNNs Reg.	87.8 %	0.634	**0.429**	0.115	0.072	0.342	0.143
5xNNs Anch.	88.0 %	0.631	0.452	**0.065**	**0.041**	**0.246**	**0.006**

Figure 10: High confidence predictions on out-of-distribution data. Mean over five runs (three for CIFAR).

Similar trends may be seen in all three tables. In terms of other data categories, all approaches predict with equal confidence on the training data, although this confidence varies widely, with anchored ensembles typically generating the most cautious forecasts. For data taken further from the training distribution, this difference widens.

One of their tests involved training a neural network (NN) to predict OOD samples that were somewhat different from the training examples and, worst case, random noise. They discovered that even if a single NN with an utterly random noise was presented, ninety-five percent of the time, it will assign this with a high probability to one of the CIFAR classes. With the use of the anchoring loss function, this drops to twelve percent.

6 CONCLUSION

This study suggested, analyzed, and tested a variation to the conventional NN ensembling process that regularizes parameters around values taken from a prior distribution and yields approximate Bayesian inference.

The methodology for the analysis is described and explained in Section 2. Sections 3 and 4 contributed to the field of study known as "Bayesian deep learning," which applies the Bayesian framework to NN parameters. The relationship between BNNs and GPs provides a valuable lens for studying a BNN's prior over functions.

The efficient learning of the posterior distribution presented a second difficulty for BNNs. The paper suggested modifying an ensemble of NNs' regularization terms such that the estimates they provide are more closely aligned with the Bayesian posterior. This is advantageous because linking ensembles of NNs with the Bayesian framework offers some assurance that their uncertainty estimates are resilient, and ensembles of NNs are simple to create and scalable.

An abstracted form of RMS was obtained under simplifying assumptions. In addition, a valid RMS variation was examined for comprehending its approximative posterior's bias. The recovery of the true posterior under two unique circumstances—perfectly correlated and extrapolation parameters—was demonstrated. The viability of using RMS on NNs, arguing that these two unique requirements are only partially present in NNs, was also questioned.

State-of-the-art performance was obtained in regression benchmarking studies on 3/10 datasets, outperforming standard approximate inference techniques. In addition, anchored ensembles were more reliable than alternative ensemble approaches on tasks involving classifying images and texts.

6.1 Alternative algorithms

Building features that enable NNs to calculate their uncertainty is still a topic of current research. However, like in many other fields of machine learning, method comparison is frequently carried out empirically through benchmark tasks using standardized benchmarks like UCI datasets.

These empirical tests effectively determine how reliable uncertainty estimates are, but they are not very informative about the effect on usability. This is crucial because it directly affects the rate of adoption by practitioners across the whole machine learning community.

An excellent illustration of this is how well-liked MC Dropout [2] is. Dropout layers can readily be included in NN designs, providing practitioners with a practical way to calculate uncertainty. Because of its ease, MC Dropout has gained widespread use even though the quality of the estimates it generates is frequently worse than alternatives.

Ensembling or VI may be a valuable option for practitioners who require more exact uncertainty estimations but are ready to put in more effort during implementation and computation. Additionally, running HMC or converting to a GP can be appropriate for individuals who require uncertainty quality above all else.

Several other Bayesian methods work on the same methodologies as RMS, like the Laplace Approximation and MFVI.

Approaches will probably shift on these axes as deep learning progresses and becomes more mature. The usage of dropout, for instance, is declining in recent visual models where it has been found to conflict with batch normalization [5] and in RL, where the inclusion of extra variance is undesirable, despite it once being ubiquitous in state-of-the-art models (about 2015). This could present a chance for novel approximations that have little effect on usability.

6.2 Future Work

Many methods used today to deal with uncertainty in NNs are drawn from those successful in more straightforward predictive models. Although the Bayesian framework has the advantage of having a solid theoretical foundation, scaling it to contemporary NNs and datasets is a difficult task.

Building ever-more-general systems are the longer-term objective of AI. Unfortunately, this will probably demand ever-larger models and datasets, so the complexity of using the Bayesian framework will only get more complex. This section speculates whether it will be possible to scale up these more basic frameworks for addressing uncertainty to more extensive and more complicated models or whether a new paradigm will be required.

Brains—the only (relatively) living example of general intelligence we have—can help us answer this question. Because of their limited experience and flawed views, they must navigate a world rife with uncertainties. According to a normative argument, managing these uncertainties was necessary for brains to evolve effectively.

This brings forth an intriguing viewpoint: possibly, learning how the brain assesses uncertainty might help us create better uncertainty-aware AI systems. Create techniques that allow more abstract BNN priors to be specified. Communicating priors through expert demonstrations, representative data sets, or straightforward hard-coded rules may be more acceptable than explicitly storing information into parameter priors. It is feasible that we will need to alter how we design methods to handle uncertainty as artificial systems become progressively more general.

7 ACKNOWLEDGMENTS

Thanks to the supervisors for helping me with this project and for their guidance.

REFERENCES

[1] Andrew Y. K. Foong, Yingzhen Li, José Miguel Hernández-Lobato, and Richard E. Turner. 2019. 'In-Between' Uncertainty in Bayesian Neural Networks. *ArXiv* abs/1906.11537 (2019).

[2] Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48* (New York, NY, USA) *(ICML'16)*. JMLR.org, 1050–1059.

[3] Z Ghahramani. 2015. Probabilistic machine learning and artificial intelligence. *Nature* 521, 7553 (May 2015), 452–459. https://doi.org/10.1038/nature14541 On Probabilistic models.

[4] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. 2016. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. https://doi.org/10.48550/ARXIV.1612.01474

[5] Jaehoon Lee, Lechao Xiao, Samuel S. Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. 2019. *Wide Neural Networks of Any Depth Evolve as Linear Models under Gradient Descent.* Curran Associates Inc., Red Hook, NY, USA.

[6] Radford M Neal. 2012. *Bayesian learning for neural networks*. Vol. 118. Springer Science & Business Media.

[7] Tim Pearce, Felix Leibfried, and Alexandra Brintrup. 2020. Uncertainty in Neural Networks: Approximately Bayesian Ensembling. In *AISTATS*. 234–244. http://proceedings.mlr.press/v108/pearce20a.html

[8] Hippolyt Ritter, Aleksandar Botev, and David Barber. 2018. A Scalable Laplace Approximation for Neural Networks. In *International Conference on Learning Representations*. https://openreview.net/forum?id=Skdvd2xAZ