On Local Posterior Structure in Deep Ensembles

Mikkel Jordahn* **Jonas Vestergaard Jensen*** **Mikkel N. Schmidt** **Michael Riis Andersen**
Technical University of Denmark
*Shared first authorship
{mikkjo,jovje}@dtu.dk

Abstract

Bayesian Neural Networks (BNNs) often improve model calibration and predictive uncertainty quantification compared to point estimators such as maximum-a-posteriori (MAP). Similarly, deep ensembles (DEs) are also known to improve calibration, and therefore, it is natural to hypothesize that deep ensembles of BNNs (DE-BNNs) should provide even further improvements. In this work, we systematically investigate this across a number of datasets, neural network architectures, and BNN approximation methods and surprisingly find that when the ensembles grow large enough, DEs consistently outperform DE-BNNs on in-distribution data. To shine light on this observation, we conduct several sensitivity and ablation studies. Moreover, we show that even though DE-BNNs outperform DEs on out-of-distribution metrics, this comes at the cost of decreased in-distribution performance. As a final contribution, we open-source the large pool of trained models to facilitate further research on this topic.

1 INTRODUCTION

Regardless of neural networks' immense popularity and successes, they are still seeing limited application in areas where uncertainty quantification and model calibration is of high importance such as in healthcare (Kompa et al., 2021; Seoni et al., 2023). This is in large part due to the widely known phenomenon that modern neural networks are often overconfident in their predictions (Nguyen et al., 2015; Guo et al., 2017). Bayesian inference applied to neural networks, i.e., Bayesian deep learning, is one area of research that in theory brings the promise of remedying poor model calibration and uncertainty quantification (Papamarkou et al., 2024).

In recent years, many different approximate Bayesian inference methods have been proposed for neural networks (MacKay, 1992; Blundell et al., 2015; Gal and Ghahramani, 2016; Maddox et al., 2019; Dusenberry et al., 2020; Daxberger et al., 2021; Harrison et al., 2024). Many of these methods have been shown to improve both in- and out-of-distribution uncertainty quantification. However, the majority of these only describe the structure around a single mode of the posterior even though neural networks posteriors are known to be highly multimodal. On the contrary, deep ensembles (DEs) (Hansen and Salamon, 1990; Lakshminarayanan et al., 2017) capture several modes of the posterior and can be interpreted as approximate Bayesian inference (Wilson and Izmailov, 2020; D'Angelo and Fortuin, 2021), yet do not capture the local structure around these modes. Nevertheless, DEs are often found to perform better than approximate inference methods that describe the local structure of a single mode. Based on the improvements provided by unimodal approximations (e.g. the Laplace approximations) and DEs separately, it is a natural hypothesis that equipping the modes of DEs with local posterior structure should further improve their performance.

In this work, we systematically investigate the effect of including local posterior structure in DEs (DE-BNNs) through approximate inference methods, namely stochastic weight averaging Gaussian (SWAG) (Maddox et al., 2019), last-layer Laplace approximation (LLLA) (Daxberger et al., 2021), and LLLA refined by normalizing flows (Kristiadi et al., 2022). In contrast to previous works, we investigate this across different datasets, neural network architectures and ensemble sizes, and surprisingly find that as we increase the number of members in the ensembles, the DEs consistently outperform the DE-BNNs on in-

Proceedings of the 28th International Conference on Artificial Intelligence and Statistics (AISTATS) 2025, Mai Khao, Thailand. PMLR: Volume 258. Copyright 2025 by the author(s).

distribution metrics. Interestingly, we also find that even though the DE-BNNs perform slightly better on out-of-distribution metrics, this often comes at the cost of worse in-distribution performance. We explore this behaviour through a range of sensitivity and ablation studies. Our contributions can be summarized as the following:

1. We show that DEs generally outperform DE-BNNs across a number of datasets, neural networks architectures and approximate BNN methods for large ensemble sizes.
2. We conduct a number of sensitivity and ablation studies to explain the different predictive performance between BNN and DE-BNNs.
3. We show that increased out-of-distribution performance in DE-BNNs often comes at an in-distribution performance cost.
4. We open-source a large class of trained BNNs for further analysis[1].

Based on these contributions and experiments, we present advice for practitioners on model choice.

2 RELATED WORK

There exists a wide range of works in Bayesian deep learning concerned with approximating the intractable posterior over neural networks. These include stochastic variational inference (SVI) methods (Blundell et al., 2015; Dusenberry et al., 2020; Shen et al., 2024; Harrison et al., 2024), Laplace approximations (MacKay, 1992; Daxberger et al., 2020, 2021), Monte Carlo dropout (Gal and Ghahramani, 2016), and stochastic gradient based approximation methods (Mandt et al., 2017; Maddox et al., 2019), all of which have been shown to improve uncertainty quantification and predictive performance over *maximum a-posterior* (MAP) estimators. Common for many of these methods is that they only approximate the posterior distribution around a single mode, which is a crude approximation considering the wide-spread knowledge that loss-landscapes of neural networks are highly multimodal.

Hamiltonian Monte Carlo (HMC) sampling is the golden standard for approximating the posterior distribution of neural networks (Neal, 1996), and can in principle explore multiple modes, but it is highly impractical to run due to computational cost. The largest HMC based BNN experiment ever done was completed in Izmailov et al. (2021) where some chains took more than *60 million epochs* to converge on CIFAR-10—and even then, some have later questioned whether the HMC chains are fully exploring the true posterior (Sharma et al., 2023). Deep ensembles (DEs) (Hansen and Salamon, 1990; Lakshminarayanan et al., 2017), on the other hand, are straightforward to implement and train, and afford large performance increases even compared to the aforementioned BNN methods. However, DEs are still only point estimates at each mode, and it seems intuitive that including local posterior structure in the DEs should further improve them.

Wilson and Izmailov (2020) and Eschenhagen et al. (2021) have previously experimented with including local posterior structure in DEs using SWAG (Maddox et al., 2019) and LLLA (Daxberger et al., 2021). In our work, we similarly investigate SWAG and LLLA and additionally the LLLA refined by normalizing flows (Kristiadi et al., 2022) for including local posterior structure in DEs but on a larger class of problems and architectures. While our results do not conflict with Wilson and Izmailov (2020) and Eschenhagen et al. (2021), we present additional findings that shine a new and different light on the current state of DE-BNNs. Firstly, we show that DEs generally perform on-par with or better than DE-BNNs on in-distribution metrics when the ensembles grow large enough. Secondly, we show that increased out-of-distribution performance observed in DE-BNNs comes at a cost of in-distribution performance.

3 BACKGROUND

In Bayesian inference, the goal is to compute the posterior distribution over model parameters $\theta \in \mathbb{R}^D$ given data $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^N$

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})}\,, \qquad (1)$$

where $p(\mathcal{D}|\theta)$ is the likelihood, $p(\theta)$ is a prior distribution over the parameters, and $p(\mathcal{D})$ is the *marginal likelihood*. Given the posterior distribution $p(\theta|\mathcal{D})$, we can make predictions for a new data point x^* via the posterior predictive distribution

$$p(y^*|x^*, \mathcal{D}) = \int_{\mathbb{R}^D} p(y^*, |x^*, \theta)p(\theta|\mathcal{D})\, d\theta\,, \qquad (2)$$

through marginalization of the model parameters θ.

In Bayesian deep learning both Eq. (1) and (2) are intractable, and we therefore resort to an approximation $q(\theta) \approx p(\theta|\mathcal{D})$ and further estimate the predictive

[1]Code and models via `https://github.com/jonasvj/OnLocalPosteriorStructureInDeepEnsembles`

distribution with Monte Carlo (MC) sampling

$$p(y^*|x^*,\mathcal{D}) \approx \frac{1}{S}\sum_{s=1}^{S} p(y^*|x^*,\theta^{(s)}), \quad \theta^{(s)} \sim q(\theta)\,. \quad (3)$$

Deep Ensembles (DEs) are a collection of K independently trained neural networks. If the loss function is chosen as $-\log p(\mathcal{D},\theta)$ and the likelihood has the form $p(\mathcal{D}|f_{\theta^{(k)}})$ where $f_{\theta^{(k)}}$ is the kth neural network, then the DE can be seen as a collection of K MAP estimates $\{\theta^{(k)}_{\mathrm{MAP}}\}_{k=1}^{K}$, i.e., K (local) maximizers of Eq. (1) w.r.t. θ. In the DE, predictions are made according to

$$p(y^*|x^*,\mathcal{D}) \approx \frac{1}{K}\sum_{k} p(y^*|x^*,\theta^{(k)}_{\mathrm{MAP}})\,. \quad (4)$$

Noting the similarity between Eq. (3) and (4), we can view DEs as a direct approximation of Eq. (2), or equivalently as an approximate posterior with the form

$$q(\theta) = \sum_{k=1}^{K} \pi_k \delta(\theta - \theta^{(k)}_{\mathrm{MAP}})\,, \quad (5)$$

where $\pi_k = \frac{1}{K}\ \forall k$ and δ is Dirac's delta distribution.

In this work, we equip DEs with local posterior structure around the MAP estimates by considering approximate Bayesian inference methods that can be computed *post-hoc* from a MAP estimate. Our approximate posteriors therefore take the form

$$q(\theta) = \sum_{k=1}^{K} \pi_k q_k(\theta|\theta^{(k)}_{\mathrm{MAP}})\,. \quad (6)$$

where $\pi_k \in [0,1]\ \forall k$ and $\sum_{k=1}^{K}\pi_k = 1$. Next, we discuss our choices for the local distributions q_k.

3.1 Last-Layer Laplace Approximation

The Laplace Approximation (LA) is a local Gaussian approximation to $p(\theta|\mathcal{D})$ centered at a MAP estimate θ_{MAP}. The LA is given by

$$q(\theta|\theta_{\mathrm{MAP}}) = \mathcal{N}(\theta|\theta_{\mathrm{MAP}}, \mathcal{H}^{-1}), \quad (7)$$

where $\mathcal{H}$ is the Hessian of the negative joint distribution, i.e.,

$$\mathcal{H} = -\nabla^2 \log p(\mathcal{D},\theta)\Big|_{\theta=\theta_{\mathrm{MAP}}}. \quad (8)$$

The size of the Hessian $\mathcal{H}$ scales quadratically in the number of network parameters D and it is therefore often necessary to approximate $\mathcal{H}$, e.g. by imposing constraints on $\mathcal{H}$, in a deep learning context (Daxberger et al., 2021). In this work, we consider the last-layer Laplace approximation (LLLA) (Kristiadi et al., 2020) in which the LA is only constructed for the parameters in the last layer of the neural network and all other parameters are fixed to their MAP estimate, i.e.,

$$q(\theta|\theta_{\mathrm{MAP}}) = \delta(\theta^{-L} - \theta^{-L}_{\mathrm{MAP}})\mathcal{N}\big(\theta^{L}|\theta^{L}_{\mathrm{MAP}}, (\mathcal{H}^{L})^{-1}\big), \quad (9)$$

where the superscript L denotes parameters in the last layer and $-L$ indexes the parameters in all earlier layers such that $\theta = \theta^{-L} \cup \theta^{L}$.

3.2 Normalizing Flows for the Last-Layer Laplace Approximation

Since neural network posteriors are complex and non-Gaussian (Garipov et al., 2018; Izmailov et al., 2021), we leverage normalizing flows (Rezende and Mohamed, 2015) to refine the Gaussian LLLA, as proposed by Kristiadi et al. (2022).

A normalizing flow creates a random variable $\theta \in \mathbb{R}^D$ with a complex probability density function by transforming a random variable θ_0 with distribution $q(\theta_0)$ using a diffeomorphism $F_\phi : \mathbb{R}^D \to \mathbb{R}^D$. F_ϕ is constructed by composing T simple diffeomorphic transformations, i.e., $F_\phi = F_{\phi_T} \circ F_{\phi_{T-1}} \circ \cdots \circ F_{\phi_1}$ with $\phi = \{\phi_t\}_{t=1}^{T}$, which ensures that the composite transformation F_ϕ is also a diffeomorphism (Papamakarios et al., 2021). Samples of θ are obtained by first sampling θ_0 and then applying F_ϕ, i.e.,

$$\theta = F_\phi(\theta_0) = F_{\phi_T}(\cdots F_{\phi_1}(\theta_0))\,, \quad \theta_0 \sim q(\theta_0)\,, \quad (10)$$

and densities are computed with the change-of-variables formula

$$\begin{aligned} q_\phi(\theta) &= q(\theta_0)|\det J_{F_\phi}(\theta_0)|^{-1} \\ &= q(\theta_0)\left|\prod_{t=1}^{T}\det J_{F_{\phi_t}}(\theta_{t-1})\right|^{-1}, \end{aligned} \quad (11)$$

with $\theta_0 = F_\phi^{-1}(\theta)$ and where $J_{F_{\phi_t}}$ is the Jacobian of F_{ϕ_t} and $\theta_t = F_{\phi_t}(\theta_{t-1})$ for $t = 1,\ldots,T$ with $\theta = \theta_T$.

When refining the LLLA with normalizing flows, we use the LLLA given in Eq. (9) as the base distribution $q(\theta_0)$ and minimize the (reverse) Kullback-Leibler (KL) divergence between $q_\phi(\theta)$ given in Eq. (11) and the true posterior $p(\theta|\mathcal{D})$ by optimizing the variational parameters ϕ, i.e.,

$$\phi^* = \arg\min_{\phi} \mathbb{KL}[q_\phi \| p] \quad (12)$$

In practice, this is done by maximizing the evidence lower bound (ELBO) w.r.t. ϕ.

3.3 Constant Stochastic Gradient Descent

The analysis of Mandt et al. (2017) shows that the iterates from stochastic gradient descent (SGD) with a constant learning rate can be seen as samples from a Gaussian approximate posterior. SWA-Gaussian (SWAG) is a practical algorithm for estimating the mean and covariance of the SGD iterates' stationary distribution. Denoting the SGD iterate after epoch m as θ_m, SWAG estimates the mean after M epochs on the objective $-\log p(\theta, \mathcal{D})$ as

$$\theta_{\text{SWA}} = \frac{1}{M} \sum_{m=1}^{M} \theta_m \,, \tag{13}$$

which is the stochastic weight averaging (SWA) (Izmailov et al., 2018) estimate of θ. The covariance matrix in SWAG is composed of a diagonal component

$$\Sigma_{\text{diag}} = \operatorname{diag}(\bar{\theta^2} - \theta_{\text{SWA}}^2) \,, \quad \bar{\theta^2} = \frac{1}{M} \sum_{m=1}^{M} \theta_m^2 \,, \tag{14}$$

(where the squares are element-wise) and a low-rank approximation of rank R using the R last iterates

$$\Sigma_{\text{low-rank}} = \frac{1}{R-1} \sum_{\substack{m \in \\ \{M-R+1,\dots,M\}}} (\theta_m - \bar{\theta}_R)(\theta_m - \bar{\theta}_R)^\top \,, \tag{15}$$

where $\bar{\theta}_R$ is the mean of the iterates from the final R epochs. The SWAG approximate posterior therefore takes the form

$$q(\theta) = \mathcal{N}\left(\theta \middle| \theta_{\text{SWA}}, \frac{1}{2}(\Sigma_{\text{diag}} + \Sigma_{\text{low-rank}})\right) . \tag{16}$$

In our experiments we initialize constant-learning-rate SGD from a MAP estimate θ_{MAP} and the SWAG posterior therefore has an implicit dependence on θ_{MAP}.

4 EXPERIMENTAL SETUP

We design and conduct a series of experiments to investigate the hypothesis that enriching deep ensembles with local posterior structure improves the predictive performance. We are particularly interested in uncertainty quantification and calibration, and therefore, we focus on the expected log predictive density (ELPD) (Gelman et al., 2014) as evaluation metric. However, we also provide results for both accuracy and expected calibration error (ECE) (Guo et al., 2017). In regression settings we report the negative mean absolute error (N-MAE) in place of accuracy. All results are presented as means ± 2 standard errors across seeds.

4.1 Datasets and Models

We conduct experiments on four real and diverse datasets: CIFAR-10 and CIFAR-100 image classification datasets (Krizhevsky and Hinton, 2009), the sentiment classification dataset SST-2 (Socher et al., 2013), and the molecular property prediction dataset QM9 (Wu et al., 2018), using the zero-point energy U_0 as target. We also conduct out-of-distribution (OOD) experiments. For the image classification models we use SVHN (Netzer et al., 2011) and CIFAR-100 data as OOD data for CIFAR-10 models and SVHN and CIFAR-10 data as OOD data for CIFAR-100 models. For SST-2 models we use the Yahoo Finance News Sentences dataset (Sadiklar, 2024) and for QM9 models we use the Tencent Alchemy dataset (Chen et al., 2019).

For the classification tasks, we use a categorical likelihood of the form $p(\mathcal{D}|\theta) = \prod_{n=1}^{N} \operatorname{cat}\big(y_n \big| f_\theta(x_n)\big)$ where $f_\theta(x)$ is a neural network that predicts the class probabilities for a given x. We use the WRN-16-4 and WRN-28-10 architectures (Zagoruyko and Komodakis, 2016) for CIFAR-10 and WRN-16-4 for CIFAR-100. For the SST-2 dataset, we use a small BERT architecture (Devlin et al., 2019) (see Appendix A). For the WRN models we use filter response normalization (FRN) (Singh and Krishnan, 2020) due to issues with batch normalization and BNNs (see Appendix B).

The molecular property prediction task from the QM9 dataset is a regression problem, and we model this using a hetereoscedastic Gaussian likelihood $p(\mathcal{D}|\theta) = \prod_{n=1}^{N} \mathcal{N}(y_n | f_\theta^{\mu}(x_n), f_\theta^{\sigma^2}(x_n))$, where f_θ is neural network that both predicts the mean and variance of y_n. For f_θ, we adapt the PaiNN graph neural network from Schütt et al. (2021) to heteroscedastic regression.

4.2 Model Training

We train each model 30 times with different initializations, yielding 30 MAP estimates of the model parameters θ for each pair of dataset and model. We then compute each of the three post-hoc approximate posteriors (SWAG, LLLA, LA-NF) for each MAP estimate. Finally, we construct DEs and DE-BNNs with Eq. (5) and (6) by randomly generating 30 model combinations without replacement for each ensemble size $K \in \{2, 5, 10, 20\}$.

The MAP estimates are obtained by minimizing the negative log-likelihood using weight decay as an implicit Gaussian-like prior. The full details of training the MAP estimates are given in Appendix A. For the BERT models, we do not leverage pre-training as the computational cost would be prohibitively expensive.

For the post-hoc approximate posteriors, we carefully tune the variance of an explicit zero-mean isotropic Gaussian prior for both the LLLA and LA-NF and the constant learning rate for SWAG using the validation ELPD as selection criterion. We use $S = 100$ sam-

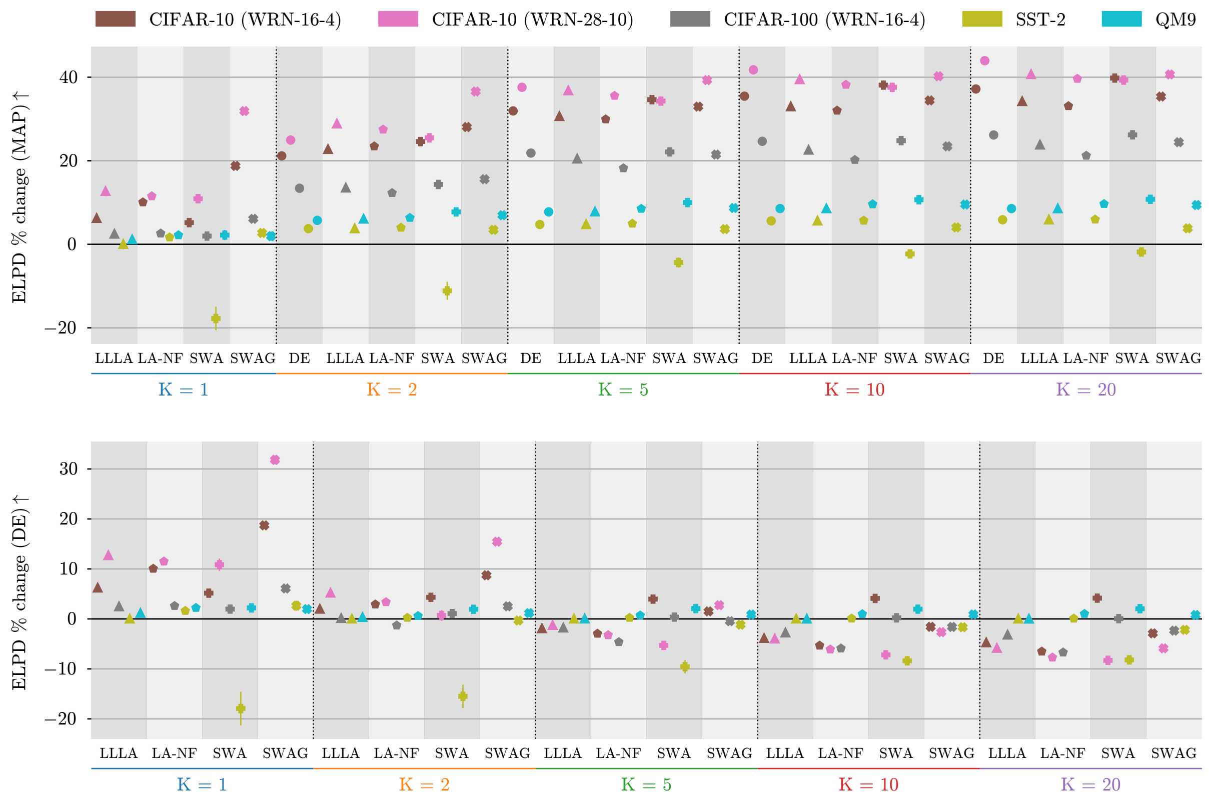

Figure 1: Top: Test ELPD change in percentage of DEs and DE-BNNs relative to MAP estimates with $K = 1$. DEs are not included here for $K = 1$ as these corresponds to the MAP estimates. Bottom: Test ELPD change in percentage of DE-BNNs relative to DEs. All points located below $y = 0$ indicate that the DE of the same ensemble size outperform the equivalently sized DE-BNN method.

ples to compute the predictive distribution in Eq. (3) during hyperparameter selection and $S = 200$ samples during model evaluation on the test set. We use $M = R = 100$ in the SWAG approximation for all models except WRN-28-10 where we use $M = R = 25$. We only use a diagonal Hessian in the LLLA for WRN-28-10 and the BERT models, but otherwise use a full Hessian. For the LA-NF we use $T \in \{1, 5, 10, 30\}$ radial flows (Rezende and Mohamed, 2015). Only results for $T = 10$ are included in the main text, and the remaining results can be found in Appendix D. Full details of computing the approximate posteriors are given in Appendix A.

Finally, we stratify the posterior samples across the K ensemble components and use uniform weights $\pi_k = \frac{1}{K}$ in Eq. (6). We investigate the effect of these choices in Section 5.2.

We also experiment with the Improved Variational Online Newton (IVON) approximate inference method proposed in Shen et al. (2024) and Monte Carlo dropout (MCDO) as proposed in Gal and Ghahramani (2016), but refer to Appendix A for training details. We only report results for these methods in Appendices D and E as these methods are not post hoc BNN methods, meaning that we cannot decouple the effect of including local posterior structure from the mode-finding (training) procedure as we can in the other post-hoc methods. However, the picture for these methods are the same as with the aforementioned methods, with the exception being MCDO on SST-2 and QM9. We discuss these results more in detail in Appendix C.

5 EXPERIMENTS & RESULTS

5.1 In-Distribution Tests

First, we investigate the in-distribution (ID) performance by evaluating ELPD on test data. Fig. 1 (top) shows the percentage change in ELPD for all combinations of models, datasets and inference methods in comparison to MAP estimated models with $K = 1$. Here we, unsurprisingly, observe that across all methods and datasets, increasing the ensemble size K significantly improves performance. Secondly, it is seen

Table 1: Average Rankings across Datasets. Ranks are computed for each seed individually yielding ranks 1-5.

	K = 1			K = 2			K = 5			K = 10			K = 20		
Inference	Acc.	ELPD	ECE	Acc.	ELPD	ECE	Acc.	ELPD	ECE	Acc.	ELPD	ECE	Acc.	ELPD	ECE
DE	3.9	4.6	4.1	3.6	4.1	2.8	3.1	2.7	**1.9**	3.0	**2.2**	**2.0**	2.9	**2.2**	**2.2**
SWA	2.2	3.5	4.1	2.2	2.9	3.8	**2.7**	2.6	2.7	**2.8**	2.6	2.3	**2.8**	2.6	2.3
SWAG	**1.6**	**1.4**	2.2	**2.0**	**1.7**	2.9	2.8	**2.4**	3.4	3.1	2.9	3.4	3.2	3.1	3.2
LLLA	4.1	3.1	2.8	3.8	3.1	**2.6**	3.4	3.5	3.3	3.2	3.5	3.5	3.2	3.4	3.7
LA-NF	3.3	2.4	**1.8**	3.3	3.2	2.9	3.0	3.8	3.7	2.9	3.7	3.7	3.0	3.6	3.6

that the BNN methods consistently yield significant performance increases for $K = 1$, except for SWA on the SST-2 dataset. We hypothesize that SWA performs poorly on SST-2 because the BERT models are trained in a low data regime without pretraining, which makes the MAP estimators highly prone to overfitting on this problem which is then exacerbated with further (SWA) gradient steps.

In Fig. 1 (bottom) we show the percentage change in ELPD for all combinations of models, datasets and inference methods in comparison to DEs with the same number of members. Here we see that already when $K = 2$, some of the DE-BNN methods perform only on-par or worse than the DE version with the same number of members. This trend continues as K is increased to 20 as almost all DE-BNNs are outperformed by DEs of the same size. It is only on the QM9 dataset where some DE-BNNs marginally outperform their DE versions.

We also note that although SWA-based DEs outperform the MAP-based DEs in several cases, these models are also only point estimates. Furthermore, SWA-based DEs are in several cases better than SWAG-based DE-BNNs (e.g., on the QM9 dataset) which points to a detrimental effect of local posterior structure since SWAG is the SWA solution equipped with local posterior structure.

To further underline that DEs generally outperform DE-BNNs for large ensemble sizes, we refer to Table 1 which shows that when we rank each inference method from 1 to 5 for each seed, dataset, and K, DEs rank the highest both for $K = 10$ and $K = 20$ on ELPD and ECE. SWA, the other point estimate-based method, comes in a close second, which further attests that local posterior structure is not necessarily beneficial. For all ensembles $K > 1$, the test accuracy is similar across inference methods and ensemble sizes. For detailed metrics including a version of Fig. 1 with percentage change in accuracy we refer to Appendices D and E.

5.2 DE-BNN Sensitivity Ablations

The previous experiment showed that DEs generally outperforms DE-BNNs for large ensembles sizes. In this section, we present a series of sensivity studies to shine more light on this observation. First, it is natural to hypothesize that DE-BNNs with large K requires a larger MC sample size for accurate results. Fig. 2 shows the effect of increasing the number of MC samples used in the posterior predictive MC approximation and the effect of stratifying samples across the members in the DE-BNNs (for $K = 20$).

Firstly, Fig. 2 (left) shows the ELPD percentage change over MAP estimated models as a function of MC samples. We observe across all approximate inference methods and datasets that the posterior predictive performance is saturated at 200 MC samples. Hence, the MC sample size does not explain the fact that DEs outperforms DE-BNNs. Secondly, Fig. 2 (right) shows the difference in ELPD percentage change for all DE-BNN methods when stratifying samples across the ensemble members. For all sample sizes the DE-BNNs perform marginally better with stratification, with the only outlier being SWAG on SST2.

Next, we investigate whether DE-BNNs can be improved by using non-uniform mixing weights. To study this, we estimate weights for each component using stacking (Yao et al., 2018) based on the predictive performance on validation data in all of the ensembles with WRN-16-4 on CIFAR-10 and CIFAR-100. We found that that the resulting mixing weights were highly uniform: Computing the normalized entropy of the discrete distributions they represent, we obtain 0.99997 on CIFAR-10 and 0.99999 on CIFAR-100 with 1 corresponding to a uniform distribution.

Since DEs can be interpreted as the limit of DE-BNNs, where the overall scale of the covariance matrices for each mixture component approaches zero, the last ablation study is designed to investigate the effect of shrinking the overall scale of covariance matrices uniformly for each component of the DE-BNNs. For this experiment, we study the WRN-16-4s for CIFAR-10 and CIFAR-100, and manipulate the covariance matrices for each mixture component by multiplying with a

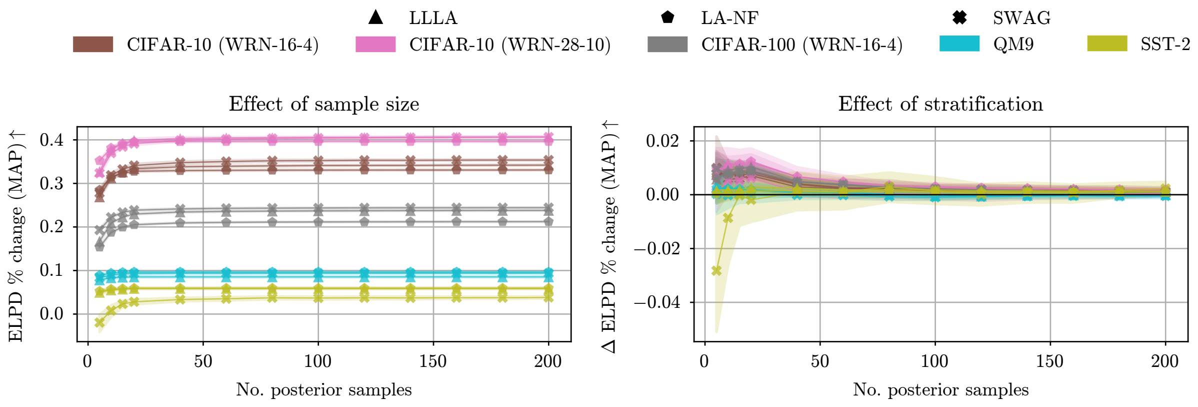

Figure 2: Left: Effect of MC samples in predictive posterior across datasets and DE-BNN methods with $K = 20$. y-axis is ELPD percentage change for a given method relative to the MAP estimate with $K = 1$. Right: Effect of stratification of samples across ensemble members for DE-BNN methods across datasets with $K = 20$. y-axis is the difference in ELPD percentage change between stratified or non-stratified samples for a given method.

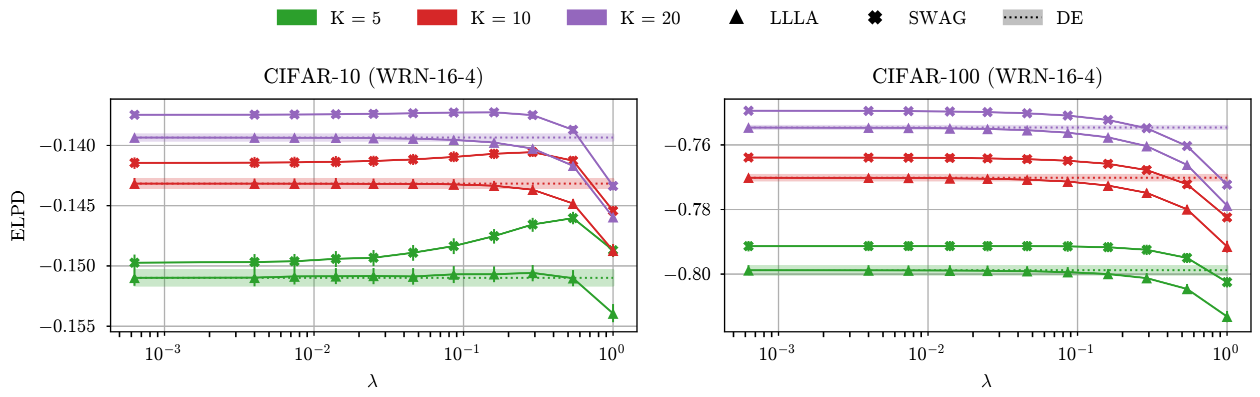

Figure 3: Test ELPD vs. covariance scaling factor λ for WRN-16-4 on CIFAR-10 (left) and CIFAR-100 (right). Dotted lines indicate DE performance for a given K.

scalar $\lambda \in [10^{-3}, 1]$. Scaling with $\lambda = 10^{-3}$ effectively reduces SWAG to SWA and LLLA to MAP. Fig. 3 shows the results for SWAG and LLLA. It is seen that for all cases, except SWAG for $K = 5$ on CIFAR-10, the scale $\lambda = 1$ is actually suboptimal and shrinking the overall scale of the covariance matrices for the DE-BNNs always results in improved ELPD. Interestingly, for SWAG on CIFAR-10, there exists values for $\lambda \in (10^{-3}, 1)$ such that the resulting λ-DE-BNNs outperforms both DEs and DE-BNNs in terms of ELPD—this is especially true for $K \in \{5, 10\}$ but the effect diminishes with the ensemble size.

5.3 Out-of-Distribution Detection

In the last experiment, we compare DEs and DE-BNNs in terms of out-of-distribution (OOD) detection. Fig. 4 visualizes the AUROC (Hanley and Mcneil, 1982) as a function of the test set ELPD (ID) for each model and dataset. It is seen that for small K, the DE-BNNs outperform the DEs, not just on in-distribution ELPD, but also on the AUROC. However, as the ensembles grow larger, i.e., for $K \in \{10, 20\}$, the DE-BNNs still outperforms the DEs w.r.t. the AUROC metric, but at the cost of reduced ID performance. Although it is observed that DE-BNNs constructed from SWAG models perform better than the MAP estimated DEs on QM9, we note that SWA DEs, another point estimate-based DE performs even better than the SWAG DE-BNNs. Lastly, the figure also shows that the best approximate inference method for OOD detection is dataset and model-dependent.

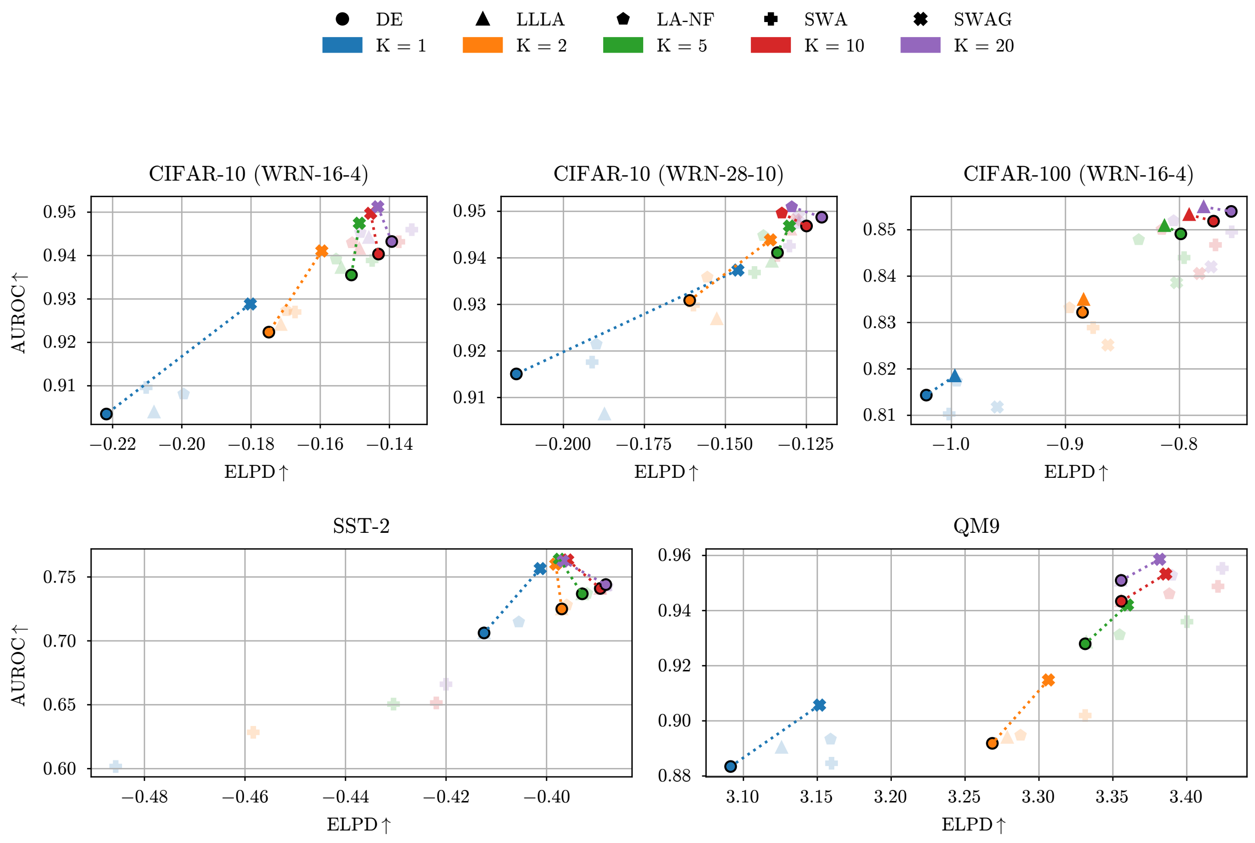

Figure 4: Out-of-distribution performance versus in-distribution test performance for DEs and DE-BNNs. DEs are highlighted with black marker edges. Lines are drawn from each DE to the DE-BNN that performs the best on the AUROC metric for a given K and dataset. Some points are faded for clarity.

6 DISCUSSION

6.1 DEs: The Practitioner's Choice

We firstly discuss whether DEs or DE-BNNs generally are recommendable to use in practice with the current state of approximate inference in BNNs. As can be seen in Fig. 1, across all of the datasets and models, we observed that for small K, DE-BNNs generally outperformed DEs both on ID and OOD metrics. Yet, as K increases to 10 and above, DEs outperformed DE-BNNs on ID metrics with DE-BNNs remaining best on OOD metrics although by a very small margin. With this finding in mind, we present two recommendations.

Firstly, if a practitioner is limited on memory for their models at prediction time, it may be beneficial to consider sub-network BNNs such as LLLA due to the limited expense of memory in such sub-network approximations and the benefit afforded by them in regimes where it not feasible to use many ensemble members. Secondly, in the regime where the practitioner is not limited by memory and can increase their ensemble member size, then DEs appear to be the better choice not only due to predictive performance, but also due to their simplicity and efficiency in terms of implementation and fitting.

Although DE-BNNs using SWAG often performed well, the memory cost and computational cost at prediction time of these models are often much larger than simply increasing the ensemble size due to how the low-rank covariance matrix is constructed and used. The memory footprint of a DE-BNN using SWAG is $\Theta(KPR)$, where K is the number of members, P is the number of parameters in the model and R is the rank of the low-rank approximation in SWAG. In contrast, the footprint for DEs is simply $\Theta(KP)$. The computational cost at prediction time is generally also much lower for DEs over SWAG-based DE-BNNs. Moreover, some of our experiments (see e.g. Fig. 1 and Fig. 3) also suggest that part of the gain observed for DE-BNNs with SWAG for larger Ks can be explained by the mean of the approximate posterior distribution being moved towards θ_{SWA} rather than a gain due to local marginalization. See Appendix D for a full overview of memory footprint, and training and prediction time cost for each model type.

6.2 On Tuning DE-BNNs

Although DE-BNNs at times outperform DEs, e.g., on OOD-detection for larger K or ID ELPD for lower K, they require careful tuning of additional hyperparameters, a contrast to the simplicity of DEs. In this work, we have carefully tuned the prior variances for LLLA and LA-NF as well as the constant learning rate used in SWAG, where especially the latter is costly. Considering the marginal benefits in OOD performance, it may be difficult to justify the computational cost and implementation burden of DE-BNNs.

The prior variance and learning rate tuning of the individual LLLA, LA-NF, and SWAG models based on the validation ELPD can be seen as a form of calibration. Even though this improves the uncertainty quantification of the individual models (see Appendix D), this might not be beneficial to the ensemble of these models, as it has been noted that an ensemble of calibrated models can lead to an under-confident ensemble (Wu and Gales, 2021). An interesting future line of work would be to investigate how to optimally combine BNNs in an ensemble instead of the näive approach taken in this work and previous DE-BNN works. Our experiment in Fig. 3 also indicates that this could be a viable direction, since the crude approach of adjusting a common scaling factor λ for each ensemble member's covariance matrix, yields a DE-BNN better than the naïvely combined DE-BNN on CIFAR-10 for SWAG and $K \in \{5, 10\}$.

7 SUMMARY & FUTURE WORK

We have investigated the effect of modelling local posterior structure in DEs (DE-BNNs) through approximate inference methods. We systematically evaluated popular approximate inference methods for BNNs and observed that they improve on MAP estimators for different data modalities. Furthermore, we have shown that DE-BNNs are superior to DEs for small ensembles, but they are mostly superfluous as the ensembles grow. For large ensembles, DEs generally have better ID performance and are much easier to train and have lower computational cost at prediction time. In line with previous works (Wilson and Izmailov, 2020; Eschenhagen et al., 2021), we observed that DE-BNNs improve OOD-detection. However, in this work we shine a new light on DE-BNNs and have shown that for large ensembles, the improvement in OOD-detection comes at the cost of decreased ID-performance.

Our results leads one to question how this counter intuitive behaviour of large DEs outperforming large DE-BNNs can persist across datasets and inference methods. Under the hypothesis that marginalization w.r.t. the true posterior distribution yields competitive predictive performance in the ID setting, there may be several explanations. Firstly, SWAG, LLLA, and LA-NF are fairly crude approximations: SWAG and LLLA both enforce Gaussianity, whereas LLLA and LA-NF only capture the posterior uncertainty for the last layer of the network, and ignore the rest. Secondly, the members of the ensembles are tuned individually, where one may hypothesise that joint tuning is likely to give better performance. Other possible explanations include poorly specified weight space priors or lack of reparametrization invariance (Roy et al., 2024). Under this new light on large DE-BNNs, investigating new multi-modal posterior approximations, both theoretically and empirically is an interesting research avenue.

Acknowledgments

The authors acknowledge support from the Novo Nordisk Foundation under grant no NNF22OC0076658 (Bayesian neural networks for molecular discovery).

References

Benjamin Kompa, Jasper Snoek, and Andrew L. Beam. Second opinion needed: communicating uncertainty in medical machine learning. *npj Digital Medicine*, 4(1):4, 2021. doi: 10.1038/s41746-020-00367-3.

Silvia Seoni, Vicnesh Jahmunah, Massimo Salvi, Prabal Datta Barua, Filippo Molinari, and U. Rajendra Acharya. Application of uncertainty quantification to artificial intelligence in healthcare: A review of last decade (2013–2023). *Computers in Biology and Medicine*, 165:107441, 2023. ISSN 0010-4825. doi: https://doi.org/10.1016/j.compbiomed.2023.107441.

Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 427–436, 2015. doi: 10.1109/CVPR.2015.7298640.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 1321–1330. JMLR.org, 2017.

Theodore Papamarkou, Maria Skoularidou, Konstantina Palla, Laurence Aitchison, Julyan Arbel, David Dunson, Maurizio Filippone, Vincent Fortuin, Philipp Hennig, José Miguel Hernández-Lobato, Aliaksandr Hubin, Alexander Immer, Theofanis Karaletsos, Mohammad Emtiyaz Khan, Agustinus Kristiadi, Yingzhen Li, Stephan Mandt, Christopher Nemeth, Michael A Osborne, Tim G. J. Rudner, David Rügamer, Yee Whye Teh, Max Welling, Andrew Gordon Wilson, and Ruqi Zhang. Position: Bayesian deep learning is needed in the age of large-scale AI. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 39556–39586. PMLR, 21–27 Jul 2024.

David JC MacKay. Bayesian interpolation. *Neural computation*, 4(3):415–447, 1992.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 1613–1622. JMLR.org, 2015.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1050–1059, New York, New York, USA, 20–22 Jun 2016. PMLR. URL `https://proceedings.mlr.press/v48/gal16.html`.

Wesley J Maddox, Pavel Izmailov, Timur Garipov, Dmitry P Vetrov, and Andrew Gordon Wilson. A simple baseline for bayesian uncertainty in deep learning. *Advances in neural information processing systems*, 32, 2019.

Michael W. Dusenberry, Ghassen Jerfel, Yeming Wen, Yi-An Ma, Jasper Snoek, Katherine Heller, Balaji Lakshminarayanan, and Dustin Tran. Efficient and scalable bayesian neural nets with rank-1 factors. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

Erik Daxberger, Agustinus Kristiadi, Alexander Immer, Runa Eschenhagen, Matthias Bauer, and Philipp Hennig. Laplace redux-effortless bayesian deep learning. *Advances in Neural Information Processing Systems*, 34:20089–20103, 2021.

James Harrison, John Willes, and Jasper Snoek. Variational bayesian last layers. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=Sx7BIiPzys`.

L.K. Hansen and Peter Salamon. Neural network ensembles. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 12:993 – 1001, 11 1990. doi: 10.1109/34.58871.

Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017.

Andrew Gordon Wilson and Pavel Izmailov. Bayesian deep learning and a probabilistic perspective of generalization. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

Francesco D'Angelo and Vincent Fortuin. Repulsive deep ensembles are bayesian. In *Neural Information Processing Systems*, 2021.

Agustinus Kristiadi, Runa Eschenhagen, and Philipp Hennig. Posterior refinement improves sample efficiency in bayesian neural networks. *Advances in Neural Information Processing Systems*, 35:30333–30346, 2022.

Yuesong Shen, Nico Daheim, Bai Cong, Peter Nickl, Gian Maria Marconi, Bazan Clement Emile Marcel Raoul, Rio Yokota, Iryna Gurevych, Daniel Cremers, Mohammad Emtiyaz Khan, and Thomas Möllenhoff. Variational learning is effective for large deep networks. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp, editors, *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 44665–44686. PMLR, 21–27 Jul 2024. URL `https://proceedings.mlr.press/v235/shen24b.html`.

Erik A. Daxberger, Eric T. Nalisnick, James Urquhart Allingham, Javier Antorán, and José Miguel Hernández-Lobato. Bayesian deep learning via subnetwork inference. In *International Conference on Machine Learning*, 2020.

Stephan Mandt, Matthew D Hoffman, and David M Blei. Stochastic gradient descent as approximate bayesian inference. *Journal of Machine Learning Research*, 18(134):1–35, 2017.

Radford M. Neal. *Bayesian Learning for Neural Networks.* Springer-Verlag, Berlin, Heidelberg, 1996. ISBN 0387947248.

Pavel Izmailov, Sharad Vikram, Matthew D Hoffman, and Andrew Gordon Gordon Wilson. What are bayesian neural network posteriors really like? In *International conference on machine learning*, pages 4629–4640. PMLR, 2021.

Mrinank Sharma, Sebastian Farquhar, Eric T. Nalisnick, and Tom Rainforth. Do bayesian neural networks need to be fully stochastic? In Francisco J. R. Ruiz, Jennifer G. Dy, and Jan-Willem van de Meent, editors, *AISTATS*, volume 206 of *Proceedings of Machine Learning Research*, pages 7694–7722. PMLR, 2023.

Runa Eschenhagen, Erik A. Daxberger, Philipp Hennig, and Agustinus Kristiadi. Mixtures of laplace approximations for improved post-hoc uncertainty in deep learning. *ArXiv*, abs/2111.03577, 2021.

Agustinus Kristiadi, Matthias Hein, and Philipp Hennig. Being bayesian, even just a bit, fixes overconfidence in relu networks. In *International conference on machine learning*, pages 5436–5446. PMLR, 2020.

Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry P Vetrov, and Andrew G Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns. *Advances in neural information processing systems*, 31, 2018.

Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *International conference on machine learning*, pages 1530–1538. PMLR, 2015.

George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64, 2021.

Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. In Ricardo Silva, Amir Globerson, and Amir Globerson, editors, *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*, 34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018, pages 876–885. Association For Uncertainty in Artificial Intelligence (AUAI), 2018. Publisher Copyright: © 34th Conference on Uncertainty in Artificial Intelligence 2018. All rights reserved.; 34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018 ; Conference date: 06-08-2018 Through 10-08-2018.

Andrew Gelman, Jessica Hwang, and Aki Vehtari. Understanding predictive information criteria for bayesian models. *Statistics and computing*, 24:997–1016, 2014.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

Zhenqin Wu, Bharath Ramsundar, Evan N Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S Pappu, Karl Leswing, and Vijay Pande. Moleculenet: a benchmark for molecular machine learning. *Chemical science*, 9(2):513–530, 2018.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Baolin Wu, Andrew Y Ng, et al. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, 2011.

Ugur Sadiklar. Yahoo finance news sentences dataset, 2024. URL `https://huggingface.co/datasets/ugursa/Yahoo-Finance-News-Sentences/tree/main`.

Guangyong Chen, Pengfei Chen, Chang-Yu Hsieh, Chee-Kong Lee, Benben Liao, Renjie Liao, Weiwen Liu, Jiezhong Qiu, Qiming Sun, Jie Tang, Richard Zemel, and Shengyu Zhang. Alchemy: A quantum chemistry dataset for benchmarking ai models. *arXiv preprint arXiv:1906.09427*, 2019.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, 2019.

Saurabh Singh and Shankar Krishnan. Filter response normalization layer: Eliminating batch dependence in the training of deep neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11237–11246, 2020.

Kristof Schütt, Oliver Unke, and Michael Gastegger. Equivariant message passing for the prediction of tensorial properties and molecular spectra. In *International Conference on Machine Learning*, pages 9377–9388. PMLR, 2021.

Yuling Yao, Aki Vehtari, Daniel Simpson, and Andrew Gelman. Using Stacking to Average Bayesian Predictive Distributions (with Discussion). *Bayesian Analysis*, 13(3):917 – 1007, 2018. doi: 10.1214/17-BA1091. URL `https://doi.org/10.1214/17-BA1091`.

J.A. Hanley and Barbara Mcneil. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143:29–36, 05 1982. doi: 10.1148/radiology.143.1.7063747.

Xixin Wu and Mark Gales. Should ensemble members be calibrated? *arXiv preprint arXiv:2101.05397*, 2021.

Hrittik Roy, Marco Miani, Carl Henrik Ek, Philipp Hennig, Marvin Pförtner, Lukas Tatzel, and Søren Hauberg. Reparameterization invariance in approximate bayesian inference, 2024. URL `https://arxiv.org/abs/2406.03334`.

Sergey Ioffe. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

Checklist

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. Yes.
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. No (although the used methods have well established time and space complexities in the original papers).
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. Yes.

2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. Not Applicable.
   (b) Complete proofs of all theoretical results. Not Applicable.
   (c) Clear explanations of any assumptions. Yes.

3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). Yes.
   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). Yes.
   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). Yes.
   (d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). Yes.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. Yes.
   (b) The license information of the assets, if applicable. Not Applicable.
   (c) New assets either in the supplemental material or as a URL, if applicable. Yes.
   (d) Information about consent from data providers/curators. Not Applicable.
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. Not Applicable.

5. If you used crowdsourcing or conducted research with human subjects, check if you include:
   (a) The full text of instructions given to participants and screenshots. Not Applicable.
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. Not Applicable.
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. Not Applicable.

On Local Posterior Structure in Deep Ensembles: Supplementary Materials

A TRAINING DETAILS

A.1 BERT Details

We use Huggingface's implementation of BERT and downscale the models. We set the hidden size to 256, have 4 layers, 8 attention heads, an intermediate size of 2048 and dropout rate to 0.5 in both the hidden and attention layers. We use a single fully connected layer as the classification layer. The remaining hyperparameters and architectural details are the default of the Hugging Face BERTConfig.

A.2 Data Splits

Across all the datasets we use the same seed (0) for splitting the training dataset into validation and training data. For CIFAR-10 and CIFAR-100 we always use 2,000 validation points. For SST-2 we use the pre-established validation dataset. For QM9 we use 10,000 points for validation and 10,831 points for testing. For CIFAR-10, CIFAR-100 and SST-2, we always test on the full pre-established test sets. For the OOD experiments, we use the test splits of the CIFAR-10, CIFAR-100, and SVHN datasets, 1,821 randomly selected datapoints from the Yahoo Finance News dataset (same size as the SST-2 test set), and 10,831 randomly selected points from the test split of the Tencent Alchemy dataset (same size as the QM9 test set).

A.3 MAP Training

For all of the MAP models trained on CIFAR-10 and CIFAR-100, we train using the SGD optimizer with a learning rate of 0.1 and cosine annealing, momentum of 0.9 and a weight decay of 5×10^{-4}. For the WRN-16-4 model we train for 100 epochs, whilst we for the large WRN-28-10 model train for 200 epochs. We always employ early-stopping based on validation ELPD. We always use data augmentation when training MAP models - in particular we use random horizontal flipping and random cropping. We refer to the GitHub repository for details on these augmentation. On both CIFAR-10 and CIFAR-100 we use a batch size of 128 during training.

For the MAP models on SST-2, we use the SGD optimizer with a weight decay of 3×10^{-3}, momentum of 0.9 and a learning rate of 0.005 with cosine annealing. We train for 100 epochs, but once again early stop based on validation ELPD. We split the full SST-2 sentences into phrases, a commonly used data augmentation for the SST-2 dataset.

For training the PaiNN MAP models on QM9, we generally follow the approach taken by Schütt et al. (2021). However, as mentioned in Section 4, we adapt the PaiNN model to be compatible with a heteroschedastic Gaussian likelihood and train using the negative log-likelihood in contrast to Schütt et al. (2021) who train with the mean squared error loss. We train the models for 650 epochs but use early stopping when we observe convergence in terms of the validation ELPD. We use the AdamW optimzer with a weight decay factor of 0.01 and an initial learning rate of 5×10^{-4} that is cosine annealed.

A.4 LLLA Training

As a general rule we do not use data augmentation during fitting of the LLLA as recommended for BNNs in Izmailov et al. (2021) - we also empirically experiment with this and find that the models perform worse when data augmentation has been used. We generally use full Hessian matrices except for the WRN-28-10 model due to memory constraints, where we instead use a diagonal covariance matrix. We fit all of the LLLA models using Monte Carlo sampling in the posterior predictive and cross validate the prior precision after fitting the Hessian. We cross-validate using a grid of size 21, with evenly spaced prior precision values in the range [-4, 4] in log space. These parameters are used across all of the datasets.

A.5 LA-NF Training

Like with the LLLA models we do not use data augmentation during fitting of the LA-NF models, and again verify emperically that this is better than including data augmentation. When fitting our flows we use radial transforms and train for 20 epochs as is done in Kristiadi et al. (2022). We use a varying number of transforms, specifically $T = \{1, 5, 10, 30\}$, and as a difference to Kristiadi et al. (2022), we cross-validate our prior precision here as well, sweeping over the values $[1, 5, 10, 20, 30, 40, 50, 70, 90, 100, 125, 150, 175, 200, 500]$, yet never observe the prior precision values to be selected on the boundaries of the grid. We use the Adam optimizer and an initial learning rate of 0.001 with cosine annealing as is done in Kristiadi et al. (2022). Finally, we employ early-stopping for the LA-NF models on QM9 based on the validation ELPD, as these were prone to overfitting.

A.6 SWA and SWAG Training

We run constant SGD from a pretrained MAP solution for 100 epochs[2] where we collect a parameter sample θ_m after each epoch. We store all these parameter samples and estimate the empirical mean and covariance of these samples using the SWAG equations given in Section 3.

We tune the learning rate by running SWAG for 21 constant learning rates in the interval $[1 \times 10^{-1}, 1 \times 10^{-4}]$[3] for each of the 30 MAP estimates. For each MAP estimate, we select the learning rate that results in the SWAG approximation with the highest validation ELPD as well as the learning rate that results in the SWA solution, θ_{SWA}, with the highest validation ELPD.

A.7 MCDO Training

We generally use the MAP hyperparameters for Monte Carlo dropout training, but cross-validate the dropout rate. This dropout rate is used both during training and inference. For all models, we sweep the dropout rate using the grid $[0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5]$. Furthermore, we include 0.6 in the grid for the WRN-28-10 models, extend the grid with $[0.6, 0.7, 0.8, 0.9]$ for the BERT models, and extend the grid with $[5 \times 10^{-7}, 1 \times 10^{-6}, 5 \times 10^{-6}, 1 \times 10^{-5}, 5 \times 10^{-5}, 1 \times 10^{-4}, 5 \times 10^{-4}, 1 \times 10^{-3}, 5 \times 10^{-3}]$ for the PaiNN models. During training, we early-stop based on the MAP solution as it is too expensive to evaluate with the full predictive distribution during training.

A.8 IVON Training

For IVON we generally use the MAP hyperparameters and the recommendations given by Shen et al. (2024). For the WRN-16-4 and WRN-28-10 models, we follow the settings given in Appendix C.2 of Shen et al. (2024). For the WRN-28-10 models, we also train for 400 epochs but saw no significant improvements compared to 200 epochs of training. For the BERT and PaiNN models, we use the MAP hyperparameters and set the IVON hyperparameters to $\beta_2 = 0.99995$, $h_0 = 0.1$, and $\lambda = N$ as recommended in Section 3 of Shen et al. (2024). As with Monte Carlo Dropout training, we early stop based on the IVON mean as it is again too expensive to evaluate with the full predictive distribution during training.

[2]except for the WRN-28-10 model where we use 25 epochs.

[3]for the PaiNN model on QM9 we use 30 learning rates in the interval $[1 \times 10^{-6}, 1 \times 10^{-10}]$.

A.9 Computational Resources

We train all models on an internal GPU-cluster with a mix of NVIDIA GPUs and only require a single GPU for each model training.

B Why Filter Norm Response and Not Batch Normalization

We here discuss why we use Filter Response Normalization (FRN) (Singh and Krishnan, 2020) and not Batch Normalization (BN) (Ioffe, 2015) in the Wide Residual Networks (WRN). Including BN is problematic in a Bayesian context both because it introduces dependencies between individual data points in the likelihood (Izmailov et al., 2021) and because the BN statistics need to be recomputed for each posterior sample (Maddox et al., 2019), significantly increasing cost at prediction time. We therefore instead use FRN in place of BN, but for completeness provide results for the same models with BN for a subset of the inference methods here in Table C.1 (the MAP, LLLA and LA-NF models) to show that the conclusions remain the same in terms of performance.

C MCDO and IVON Results & Discussion

In this section, we discuss the IVON and MCDO results. For in-distribution ELPD performance we refer to Table D.3 and Figure E.1. For both IVON and MC-Dropout (MCDO), we generally observe the same pattern as with the remaining methods. For small K, DE-BNNs constructed from models trained with these methods outperform DEs, but as the ensembles grow large enough, the DEs outperform DE-BNNs on in-distribution metrics.

We note that for the MCDO models on QM9 and SST-2, the DE-BNNs outperform DEs even for large Ks, but this may be explained by the effect of better tuned regularization, due to increased dropout rates and more regularization in these models, rather than local marginalization. We also note that IVON models perform poorly even for small K for a number of models, including the large WRN-28-10 on CIFAR-10 and the SST-2 and QM9 models. We hypothesize that this performance gap likely could be improved by more extensive hyperparameter tuning of the IVON optimizer. However, such hyperparameter tuning is highly expensive due to IVON training not being performed post-hoc, but rather when training from scratch. Please also note that the IVON models on QM9 have been left out of Figure E.1 because these models have not converged to a meaningful solution as evident in Table D.3.

With regards to OOD performance we refer to Figure E.3. Here we observe that consistent with the other methods, the DE-BNNs constructed with well-tuned IVON and MCDO models consistently outperform the DEs as previously noted.

Table C.1: WRN Results with Batch Normalization

		CIFAR-10 (WRN-16-4)			CIFAR-100 (WRN-16-4)		
K	Inference	Acc.↑	ELPD↑	ECE↓	Acc.↑	ELPD↑	ECE↓
1	DE	**0.947**	-0.185	0.023	**0.760**	-0.949	0.082
	LLLA	**0.947**	-0.174	0.014	**0.760**	-0.885	0.029
	LLLA-NF-1	**0.947**	-0.171	0.010	**0.760**	**-0.884**	0.022
	LLLA-NF-5	**0.947**	**-0.169**	**0.008**	**0.760**	-0.886	**0.019**
	LLLA-NF-10	**0.947**	**-0.169**	**0.008**	**0.760**	-0.886	**0.019**
	LLLA-NF-30	**0.947**	**-0.169**	0.009	**0.760**	-0.893	0.027
5	DE	**0.957**	**-0.134**	**0.005**	0.800	**-0.719**	**0.019**
	LLLA	**0.957**	**-0.134**	0.008	0.801	-0.736	0.070
	LLLA-NF-1	**0.957**	-0.135	0.012	**0.802**	-0.753	0.087
	LLLA-NF-5	**0.957**	-0.137	0.015	**0.802**	-0.750	0.082
	LLLA-NF-10	**0.957**	-0.137	0.015	**0.802**	-0.750	0.082
	LLLA-NF-30	**0.957**	-0.137	0.014	**0.802**	-0.765	0.093
10	DE	**0.959**	**-0.128**	**0.005**	0.806	**-0.688**	**0.025**
	LLLA	**0.959**	-0.131	0.011	0.807	-0.717	0.078
	LLLA-NF-1	**0.959**	-0.132	0.015	**0.808**	-0.737	0.096
	LLLA-NF-5	**0.959**	-0.134	0.017	**0.808**	-0.733	0.091
	LLLA-NF-10	**0.959**	-0.133	0.017	**0.808**	-0.733	0.091
	LLLA-NF-30	**0.959**	-0.134	0.017	0.807	-0.749	0.102
20	DE	**0.959**	**-0.125**	**0.005**	0.810	**-0.672**	**0.028**
	LLLA	**0.959**	-0.128	0.012	0.810	-0.708	0.084
	LLLA-NF-1	**0.959**	-0.130	0.015	**0.811**	-0.729	0.102
	LLLA-NF-5	**0.959**	-0.132	0.018	**0.811**	-0.724	0.096
	LLLA-NF-10	**0.959**	-0.131	0.018	**0.811**	-0.723	0.096
	LLLA-NF-30	**0.959**	-0.131	0.017	**0.811**	-0.740	0.107

D ADDITIONAL TABLES

Table D.2: Average Rankings Across Datasets Including All Flow Lengths.

	K = 1			K = 2			K = 5			K = 10			K = 20		
Inference	Acc.	ELPD	ECE	Acc.	ELPD	ECE	Acc.	ELPD	ECE	Acc.	ELPD	ECE	Acc.	ELPD	ECE
DE	7.0	8.9	8.2	6.0	7.4	5.4	4.9	4.2	**3.1**	4.8	**3.5**	**3.3**	4.6	**3.7**	3.6
SWA	3.4	7.1	8.6	3.3	5.3	7.0	**4.5**	4.3	4.4	**4.7**	4.2	3.4	**4.5**	4.2	**3.3**
SWAG	**2.3**	**2.8**	4.4	**3.0**	3.0	5.0	**4.5**	3.8	5.7	5.3	4.4	5.7	5.4	4.7	5.4
LLLA	7.3	6.5	6.2	6.1	5.7	4.4	5.3	5.5	5.2	5.3	5.4	5.5	5.0	5.5	6.0
LLLA-NF-1	5.9	4.9	4.7	6.2	6.5	5.1	5.9	6.5	5.4	5.2	6.0	5.8	5.2	5.2	5.5
LLLA-NF-5	5.5	5.0	4.2	5.6	5.6	5.1	5.2	6.3	5.8	4.8	6.2	5.9	5.2	6.1	5.5
LLLA-NF-10	5.5	4.9	**3.9**	5.1	6.2	5.5	4.8	6.8	6.6	**4.7**	6.7	6.3	4.8	6.7	6.2
LLLA-NF-30	5.8	5.5	**3.9**	5.3	5.7	5.2	4.8	6.6	6.2	4.8	7.0	6.4	4.9	7.1	6.4
IVON	7.4	6.6	6.7	8.4	7.0	8.2	8.7	7.6	8.4	8.8	7.8	8.4	8.6	7.9	8.5
MCDO	4.9	**2.8**	4.3	5.9	**2.7**	**4.1**	6.5	**3.5**	4.1	6.5	3.7	4.3	6.8	3.9	4.6

Table D.3: Test Metric Means.

		CIFAR-10 (WRN-16-4)			CIFAR-10 (WRN-28-10)			CIFAR-100 (WRN-16-4)			SST-2			QM9		
K	Inference	Acc.↑	ELPD↑	ECE↓	Acc.↑	ELPD↑	ECE↓	Acc.↑	ELPD↑	ECE↓	Acc.↑	ELPD↑	ECE↓	MAE↓	ELPD↑	ECE↓
	DE	0.938	-0.222	0.032	0.947	-0.215	0.033	0.736	-1.022	0.066	0.818	-0.412	0.033	8.859	3.091	0.051
	SWA	**0.942**	-0.210	0.031	**0.953**	-0.191	0.030	0.739	-1.002	0.058	0.811	-0.486	0.089	**8.115**	3.160	0.041
	SWAG	0.941	-0.180	0.009	0.952	**-0.146**	**0.006**	**0.744**	-0.960	0.020	**0.825**	**-0.401**	0.031	8.145	3.152	0.049
1	LLLA	0.938	-0.208	0.018	0.947	-0.187	0.016	0.735	-0.997	0.025	0.818	-0.412	0.033	8.914	3.126	0.047
	LA-NF	0.938	-0.199	0.013	0.947	-0.190	0.015	0.735	-0.995	**0.013**	0.823	-0.406	0.027	8.553	3.159	**0.034**
	IVON	0.940	**-0.179**	**0.007**	0.927	-0.225	0.032	0.738	-0.953	0.017	0.655	-0.578	0.057	2540.630	-7.496	0.166
	MCDO	0.940	-0.181	0.010	0.952	-0.153	0.014	0.733	**-0.942**	0.018	0.819	-0.404	**0.024**	8.339	**3.165**	0.045
	DE	0.947	-0.175	0.014	0.954	-0.161	0.015	0.761	-0.885	**0.019**	0.827	-0.397	**0.024**	7.306	3.268	**0.081**
	SWA	**0.949**	-0.167	0.014	**0.957**	-0.160	0.019	0.763	-0.876	0.020	0.818	-0.458	0.074	**6.842**	**3.331**	0.082
	SWAG	0.948	-0.160	0.009	0.956	**-0.136**	**0.005**	**0.767**	**-0.863**	0.037	0.827	-0.398	0.032	6.869	3.306	0.089
2	LLLA	0.947	-0.171	0.009	0.954	-0.153	0.007	0.761	-0.884	0.032	0.827	-0.397	0.025	7.359	3.279	0.082
	LA-NF	0.947	-0.170	0.009	0.954	-0.156	0.009	0.761	-0.896	0.054	**0.829**	-0.396	**0.024**	7.358	3.288	0.082
	IVON	0.947	**-0.159**	0.011	0.934	-0.212	0.041	0.760	-0.865	0.030	0.691	-0.595	0.121	2536.273	-7.276	0.165
	MCDO	0.946	-0.162	**0.008**	0.956	-0.138	0.008	0.754	-0.866	0.032	0.828	**-0.391**	0.026	6.953	3.329	0.082
	DE	0.951	-0.151	0.008	0.958	-0.134	0.008	0.782	-0.799	**0.034**	**0.832**	-0.393	0.022	6.229	3.331	0.116
	SWA	**0.953**	**-0.145**	**0.007**	0.958	-0.141	0.014	0.781	**-0.796**	0.035	0.819	-0.430	0.058	**5.969**	**3.400**	**0.112**
	SWAG	0.952	-0.149	0.015	0.958	-0.130	0.008	**0.784**	-0.802	0.063	0.826	-0.397	0.031	6.012	3.360	0.118
5	LLLA	0.951	-0.154	0.012	0.958	-0.136	0.012	0.782	-0.813	0.064	**0.832**	-0.393	0.022	6.271	3.332	0.116
	LA-NF	0.952	-0.155	0.018	**0.959**	-0.138	0.014	0.781	-0.836	0.086	0.831	-0.392	**0.019**	6.215	3.355	0.114
	IVON	0.952	-0.150	0.019	0.938	-0.205	0.049	0.774	-0.814	0.054	0.766	-0.560	0.156	2539.758	-7.223	0.165
	MCDO	0.950	-0.152	0.012	0.958	**-0.128**	**0.005**	0.768	-0.820	0.056	0.831	**-0.384**	0.029	6.022	3.377	0.116
	DE	0.953	-0.143	0.009	**0.960**	**-0.125**	0.007	0.788	-0.770	**0.044**	0.835	-0.389	0.022	5.748	3.356	0.127
	SWA	**0.954**	**-0.137**	**0.006**	0.959	-0.134	0.012	0.788	**-0.768**	**0.044**	0.822	-0.422	0.052	5.612	**3.421**	**0.123**
	SWAG	0.953	-0.145	0.018	0.958	-0.128	0.008	**0.791**	-0.782	0.073	0.828	-0.396	0.033	5.645	3.386	0.127
10	LLLA	0.953	-0.149	0.016	**0.960**	-0.130	0.015	0.788	-0.792	0.077	0.835	-0.389	0.022	5.780	3.356	0.127
	LA-NF	0.953	-0.151	0.021	**0.960**	-0.133	0.017	0.788	-0.816	0.097	0.834	-0.389	**0.018**	5.722	3.388	0.125
	IVON	0.953	-0.146	0.021	0.940	-0.203	0.051	0.779	-0.795	0.063	0.797	-0.550	0.182	2537.032	-7.140	0.165
	MCDO	0.951	-0.149	0.014	0.959	**-0.125**	**0.004**	0.773	-0.807	0.065	**0.836**	**-0.381**	0.034	**5.584**	3.403	0.127
	DE	0.954	-0.139	0.009	**0.961**	**-0.120**	0.007	0.792	-0.755	**0.049**	**0.837**	-0.388	0.024	5.495	3.356	0.134
	SWA	**0.955**	**-0.134**	**0.005**	0.960	-0.130	0.011	0.792	**-0.754**	0.050	0.821	-0.420	0.051	**5.410**	**3.424**	**0.129**
	SWAG	0.954	-0.143	0.018	0.958	-0.127	0.009	**0.794**	-0.772	0.079	0.826	-0.397	0.032	5.438	3.382	0.134
20	LLLA	0.954	-0.146	0.017	**0.961**	-0.127	0.017	0.792	-0.779	0.082	**0.837**	-0.388	0.023	5.517	3.355	0.134
	LA-NF	0.954	-0.148	0.022	**0.961**	-0.130	0.018	0.791	-0.805	0.103	0.835	-0.388	**0.018**	5.470	3.390	0.132
	IVON	0.954	-0.145	0.023	0.940	-0.203	0.052	0.782	-0.785	0.067	0.802	-0.543	0.183	2540.415	-7.130	0.165
	MCDO	0.952	-0.147	0.015	0.959	-0.124	**0.004**	0.776	-0.797	0.068	**0.837**	**-0.381**	0.036	**5.410**	3.395	0.134

Table D.4: Standard Errors of Test Metric Means.

		CIFAR-10 (WRN-16-4)			CIFAR-10 (WRN-28-10)			CIFAR-100 (WRN-16-4)			SST-2			QM9		
K	Inference	Acc.	ELPD	ECE	Acc.	ELPD	ECE	Acc.	ELPD	ECE	Acc.	ELPD	ECE	MAE	ELPD	ECE
	DE	0.000	0.001	0.000	0.001	0.001	0.000	0.001	0.002	0.001	0.002	0.003	0.002	0.123	0.018	0.004
	SWA	0.000	0.001	0.000	0.000	0.001	0.000	0.001	0.002	0.001	0.002	0.006	0.004	0.087	0.019	0.001
	SWAG	0.000	0.001	0.000	0.000	0.001	0.000	0.001	0.002	0.000	0.001	0.002	0.002	0.088	0.018	0.002
1	LLLA	0.000	0.001	0.001	0.001	0.001	0.000	0.001	0.002	0.001	0.002	0.003	0.002	0.121	0.014	0.003
	LA-NF	0.000	0.001	0.000	0.001	0.001	0.000	0.001	0.002	0.001	0.001	0.002	0.002	0.098	0.014	0.002
	IVON	0.001	0.003	0.000	0.001	0.003	0.001	0.001	0.006	0.002	0.026	0.022	0.003	5.669	0.087	0.001
	MCDO	0.000	0.001	0.001	0.000	0.001	0.000	0.001	0.002	0.002	0.001	0.002	0.002	0.104	0.019	0.003
	DE	0.000	0.001	0.000	0.000	0.001	0.000	0.001	0.001	0.000	0.001	0.001	0.002	0.074	0.012	0.001
	SWA	0.000	0.001	0.000	0.000	0.001	0.000	0.001	0.002	0.000	0.001	0.005	0.003	0.057	0.010	0.001
	SWAG	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.000	0.001	0.001	0.002	0.057	0.010	0.001
2	LLLA	0.000	0.001	0.000	0.000	0.001	0.000	0.001	0.001	0.001	0.001	0.001	0.002	0.072	0.011	0.001
	LA-NF	0.000	0.001	0.000	0.000	0.001	0.000	0.000	0.001	0.001	0.001	0.001	0.001	0.076	0.012	0.002
	IVON	0.000	0.001	0.000	0.001	0.001	0.001	0.000	0.001	0.001	0.024	0.015	0.010	3.070	0.057	0.000
	MCDO	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.002	0.001	0.001	0.001	0.002	0.043	0.009	0.001
	DE	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.000	0.001	0.001	0.001	0.030	0.006	0.001
	SWA	0.000	0.000	0.000	0.000	0.001	0.000	0.001	0.001	0.001	0.001	0.003	0.002	0.024	0.005	0.001
	SWAG	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.000	0.001	0.001	0.001	0.025	0.006	0.001
5	LLLA	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.001	0.001	0.001	0.001	0.030	0.006	0.001
	LA-NF	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.001	0.001	0.001	0.001	0.031	0.007	0.001
	IVON	0.000	0.000	0.000	0.000	0.001	0.000	0.000	0.001	0.000	0.013	0.012	0.007	2.418	0.040	0.000
	MCDO	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.001	0.001	0.001	0.001	0.040	0.006	0.001
	DE	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.000	0.001	0.022	0.005	0.000
	SWA	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.000	0.001	0.002	0.002	0.020	0.004	0.000
	SWAG	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.001	0.001	0.020	0.004	0.000
10	LLLA	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.001	0.001	0.000	0.001	0.022	0.005	0.000
	LA-NF	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.000	0.000	0.001	0.019	0.004	0.000
	IVON	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.002	0.006	0.005	1.527	0.021	0.000
	MCDO	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.001	0.001	0.000	0.001	0.030	0.004	0.000
	DE	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.011	0.002	0.000
	SWA	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.001	0.010	0.001	0.000
	SWAG	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.001	0.001	0.009	0.002	0.000
20	LLLA	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.010	0.002	0.000
	LA-NF	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.009	0.002	0.000
	IVON	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.003	0.003	0.807	0.013	0.000
	MCDO	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.000	0.000	0.000	0.001	0.015	0.002	0.000

Table D.5: Test Metric Means For All Flow Lengths.

		CIFAR-10 (WRN-16-4)			CIFAR-10 (WRN-28-10)			CIFAR-100 (WRN-16-4)			SST-2			QM9		
K	Inference	Acc.↑	ELPD↑	ECE↓	Acc.↑	ELPD↑	ECE↓	Acc.↑	ELPD↑	ECE↓	Acc.↑	ELPD↑	ECE↓	MAE↓	ELPD↑	ECE↓
	LLLA	**0.938**	-0.208	0.018	0.947	**-0.187**	0.016	0.735	-0.997	0.025	0.818	-0.412	0.033	8.914	3.126	0.047
	LLLA-NF-1	**0.938**	-0.206	0.018	0.947	**-0.187**	0.016	**0.736**	**-0.995**	0.014	0.822	**-0.405**	**0.025**	8.590	3.155	0.035
1	LLLA-NF-5	**0.938**	-0.201	0.014	**0.948**	-0.189	**0.015**	0.735	**-0.995**	**0.013**	0.822	**-0.405**	0.026	8.577	3.158	0.034
	LLLA-NF-10	**0.938**	**-0.199**	0.013	0.947	-0.190	**0.015**	0.735	**-0.995**	**0.013**	**0.823**	-0.406	0.027	8.553	3.159	0.034
	LLLA-NF-30	**0.938**	**-0.199**	**0.012**	0.947	-0.190	**0.015**	0.735	-1.000	0.020	0.822	-0.406	0.027	**8.535**	**3.161**	**0.032**
	LLLA	**0.947**	-0.171	**0.009**	**0.954**	**-0.153**	**0.007**	0.761	**-0.884**	**0.032**	0.827	-0.397	0.025	7.359	3.279	0.082
	LLLA-NF-1	**0.947**	-0.171	**0.009**	**0.954**	-0.154	0.011	**0.762**	-0.892	0.049	0.828	-0.397	0.023	9.618	3.149	0.087
2	LLLA-NF-5	**0.947**	**-0.170**	**0.009**	**0.954**	-0.155	0.010	0.761	-0.896	0.053	**0.829**	**-0.395**	**0.021**	7.375	3.286	0.084
	LLLA-NF-10	**0.947**	**-0.170**	**0.009**	**0.954**	-0.156	0.009	0.761	-0.896	0.054	**0.829**	-0.396	0.024	7.358	3.288	0.082
	LLLA-NF-30	**0.947**	**-0.170**	**0.009**	**0.954**	-0.155	0.009	0.761	-0.906	0.065	**0.829**	-0.396	0.024	**7.170**	**3.316**	**0.079**
	LLLA	0.951	**-0.154**	**0.012**	0.958	**-0.136**	**0.012**	**0.782**	**-0.813**	**0.064**	**0.832**	-0.393	0.022	6.271	3.332	0.116
	LLLA-NF-1	**0.952**	**-0.154**	0.013	0.958	-0.137	0.014	**0.782**	-0.831	0.083	0.830	-0.393	0.020	7.232	3.289	0.116
5	LLLA-NF-5	**0.952**	-0.155	0.016	**0.959**	-0.137	0.013	0.781	-0.835	0.086	0.831	**-0.392**	**0.019**	6.246	3.349	0.114
	LLLA-NF-10	**0.952**	-0.155	0.018	**0.959**	-0.138	0.014	0.781	-0.836	0.086	0.831	**-0.392**	**0.019**	6.215	3.355	0.114
	LLLA-NF-30	**0.952**	-0.155	0.018	**0.959**	-0.138	0.014	0.780	-0.849	0.097	0.831	**-0.392**	**0.019**	**6.136**	**3.379**	**0.111**
	LLLA	**0.953**	**-0.149**	**0.016**	**0.960**	**-0.130**	**0.015**	**0.788**	**-0.792**	**0.077**	**0.835**	**-0.389**	0.022	5.780	3.356	0.127
	LLLA-NF-1	**0.953**	**-0.149**	0.017	**0.960**	-0.131	0.016	**0.788**	-0.812	0.096	0.834	**-0.389**	**0.018**	5.996	3.353	0.127
10	LLLA-NF-5	**0.953**	-0.150	0.020	**0.960**	-0.131	0.016	**0.788**	-0.815	0.097	0.834	**-0.389**	**0.018**	5.756	3.380	0.125
	LLLA-NF-10	**0.953**	-0.151	0.021	**0.960**	-0.133	0.017	**0.788**	-0.816	0.097	0.834	**-0.389**	**0.018**	5.722	3.388	0.125
	LLLA-NF-30	**0.953**	-0.151	0.021	**0.960**	-0.133	0.017	0.787	-0.829	0.107	0.834	**-0.389**	**0.018**	**5.704**	**3.405**	**0.123**
	LLLA	**0.954**	**-0.146**	**0.017**	**0.961**	**-0.127**	**0.017**	**0.792**	**-0.779**	**0.082**	**0.837**	**-0.388**	0.023	5.517	3.355	0.134
	LLLA-NF-1	**0.954**	**-0.146**	0.018	**0.961**	-0.128	**0.017**	**0.792**	-0.802	0.102	0.836	**-0.388**	0.019	5.565	3.364	0.134
20	LLLA-NF-5	**0.954**	-0.148	0.021	**0.961**	-0.128	**0.017**	0.791	-0.805	0.103	0.835	**-0.388**	**0.018**	5.490	3.383	0.132
	LLLA-NF-10	**0.954**	-0.148	0.022	**0.961**	-0.130	0.018	0.791	-0.805	0.103	0.835	**-0.388**	**0.018**	5.470	3.390	0.132
	LLLA-NF-30	**0.954**	-0.149	0.023	**0.961**	-0.130	0.018	0.791	-0.820	0.113	0.835	**-0.388**	**0.018**	**5.462**	**3.406**	**0.130**

Table D.6: Standard Errors of Test Metric Means For All Flow Lenghts.

		CIFAR-10 (WRN-16-4)			CIFAR-10 (WRN-28-10)			CIFAR-100 (WRN-16-4)			SST-2			QM9		
K	Inference	Acc.	ELPD	ECE	Acc.	ELPD	ECE	Acc.	ELPD	ECE	Acc.	ELPD	ECE	MAE	ELPD	ECE
	LLLA	0.000	0.001	0.001	0.001	0.001	0.000	0.001	0.002	0.001	0.002	0.003	0.002	0.121	0.014	0.003
	LLLA-NF-1	0.000	0.001	0.000	0.001	0.001	0.000	0.001	0.002	0.001	0.001	0.002	0.002	0.100	0.014	0.002
1	LLLA-NF-5	0.000	0.001	0.000	0.001	0.001	0.000	0.001	0.002	0.000	0.001	0.002	0.002	0.098	0.014	0.002
	LLLA-NF-10	0.000	0.001	0.000	0.001	0.001	0.000	0.001	0.002	0.001	0.001	0.002	0.002	0.098	0.014	0.002
	LLLA-NF-30	0.000	0.001	0.000	0.001	0.001	0.000	0.001	0.002	0.001	0.001	0.002	0.002	0.096	0.015	0.001
	LLLA	0.000	0.001	0.000	0.000	0.001	0.000	0.001	0.001	0.001	0.001	0.001	0.002	0.072	0.011	0.001
	LLLA-NF-1	0.000	0.001	0.000	0.000	0.001	0.000	0.000	0.001	0.002	0.001	0.001	0.002	0.627	0.030	0.005
2	LLLA-NF-5	0.000	0.001	0.000	0.000	0.001	0.000	0.000	0.001	0.001	0.001	0.001	0.001	0.091	0.012	0.002
	LLLA-NF-10	0.000	0.001	0.000	0.000	0.001	0.000	0.000	0.001	0.001	0.001	0.001	0.001	0.076	0.012	0.002
	LLLA-NF-30	0.000	0.001	0.000	0.000	0.001	0.000	0.000	0.001	0.001	0.001	0.001	0.001	0.060	0.009	0.001
	LLLA	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.001	0.001	0.001	0.001	0.030	0.006	0.001
	LLLA-NF-1	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.001	0.001	0.001	0.001	0.278	0.013	0.002
5	LLLA-NF-5	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.000	0.001	0.001	0.001	0.030	0.007	0.001
	LLLA-NF-10	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.001	0.001	0.001	0.001	0.031	0.007	0.001
	LLLA-NF-30	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.001	0.001	0.001	0.001	0.023	0.006	0.001
	LLLA	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.001	0.001	0.000	0.001	0.022	0.005	0.000
	LLLA-NF-1	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.001	0.001	0.000	0.001	0.068	0.005	0.000
10	LLLA-NF-5	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.000	0.001	0.022	0.004	0.000
	LLLA-NF-10	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.000	0.000	0.001	0.019	0.004	0.000
	LLLA-NF-30	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.001	0.000	0.000	0.001	0.020	0.004	0.000
	LLLA	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.010	0.002	0.000
	LLLA-NF-1	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.018	0.002	0.000
20	LLLA-NF-5	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.009	0.002	0.000
	LLLA-NF-10	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.009	0.002	0.000
	LLLA-NF-30	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.009	0.002	0.000

Table D.7: Comparison of the Probit Approximation and Monte Carlo sampling in LLLA. The CV column denotes the approximation used during cross-validation (fitting) and the Test column denotes the approximation used during inference/test time.

			CIFAR-10 (WRN-16-4)		
K	CV	Test	Acc.$\uparrow$	ELPD$\uparrow$	ECE$\downarrow$
	MC	MC	**0.938**	-0.208	0.018
1	MC	Probit	**0.938**	-0.358	0.146
	Probit	Probit	**0.938**	**-0.201**	**0.016**
	MC	MC	**0.947**	-0.171	**0.009**
2	MC	Probit	0.946	-0.300	0.133
	Probit	Probit	0.946	**-0.170**	**0.009**
	MC	MC	0.951	**-0.154**	**0.012**
5	MC	Probit	0.951	-0.295	0.149
	Probit	Probit	**0.952**	**-0.154**	0.017
	MC	MC	**0.953**	**-0.149**	**0.016**
10	MC	Probit	**0.953**	-0.313	0.170
	Probit	Probit	**0.953**	**-0.149**	0.020
	MC	MC	**0.954**	**-0.146**	**0.017**
20	MC	Probit	**0.954**	-0.307	0.169
	Probit	Probit	**0.954**	-0.147	0.022

Table D.8: Memory, Prediction Time, and Training Cost in Big O Notation. In the table, P_L is the number of parameters in the last layer, P_{-L} is the number of parameters in the remaining layers, $P = P_{-L} + P_L$ is the total number of parameters, K is the number of ensemble members, R is the rank of the low-rank SWAG approximation, S is the number of samples used in the MC integration, E is the number of epochs required during training to reach a mode, F is the number of radial flows in LA-NF, and E_V is the number of variational training epochs for LA-NF. Single mode approximations are special cases of the presented complexities where $K = 1$.

Method	Memory	Prediction Time	Training Cost
DE	KP	KP	$KNPE$
Multi-SWAG	KPR	SPR	$KNP(E+R)$
MoLA	$K(P_{-L}+P_L^2)$	$KP_{-L}+SP_L^2$	$KN(P_{-L}+P_L^3)+KNPE$
Multi-IVON	KP	SP	$KNPE$
Multi-MCDO	KP	SP	$KNPE$
MoLA-NF	$K(P_{-L}+P_L^2+FP_L)$	$KP_{-L}+S(P_L^2+FP_L)$	$KN(P_{-L}+P_L^3+FP_LE_V)+KNPE$

E ADDITIONAL FIGURES

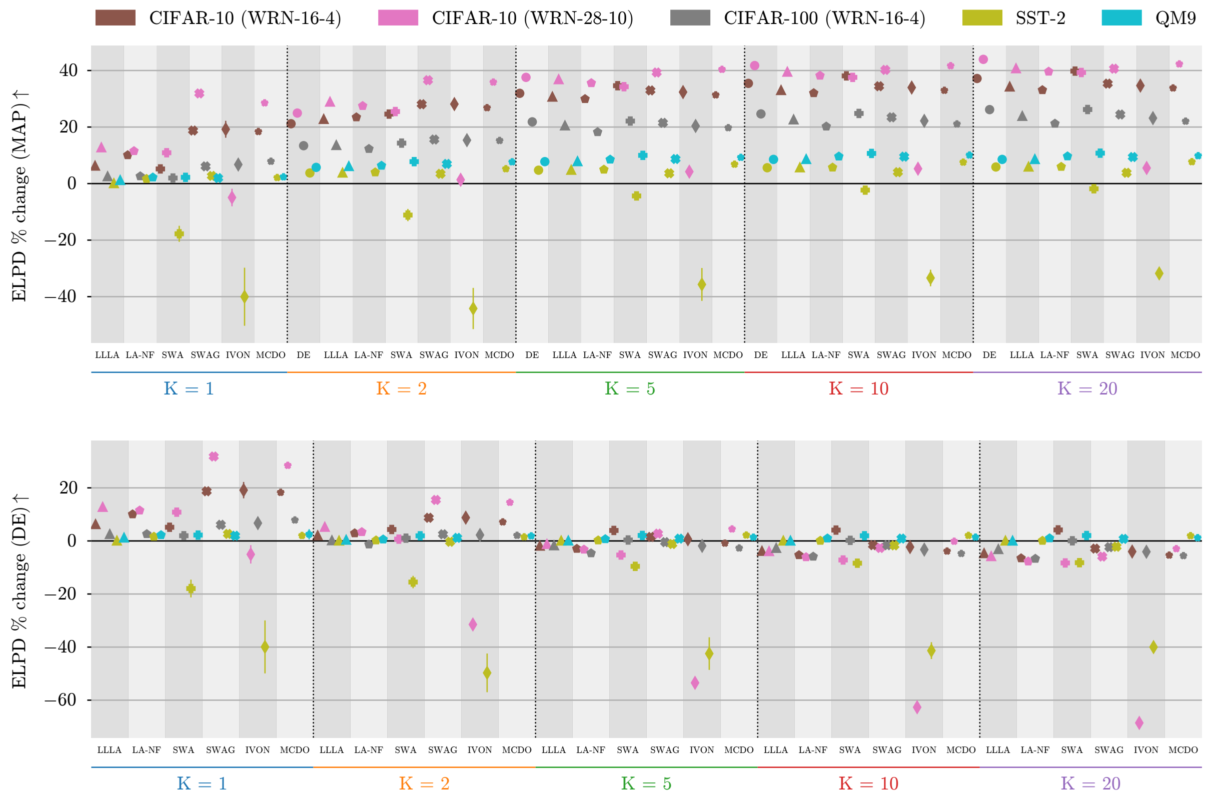

Figure E.1: ELPD % Change for All Inference Methods and Datasets Excluding IVON on QM9. (top) is versus MAP ($K = 1$) and (bottom) is versus DE with same K.

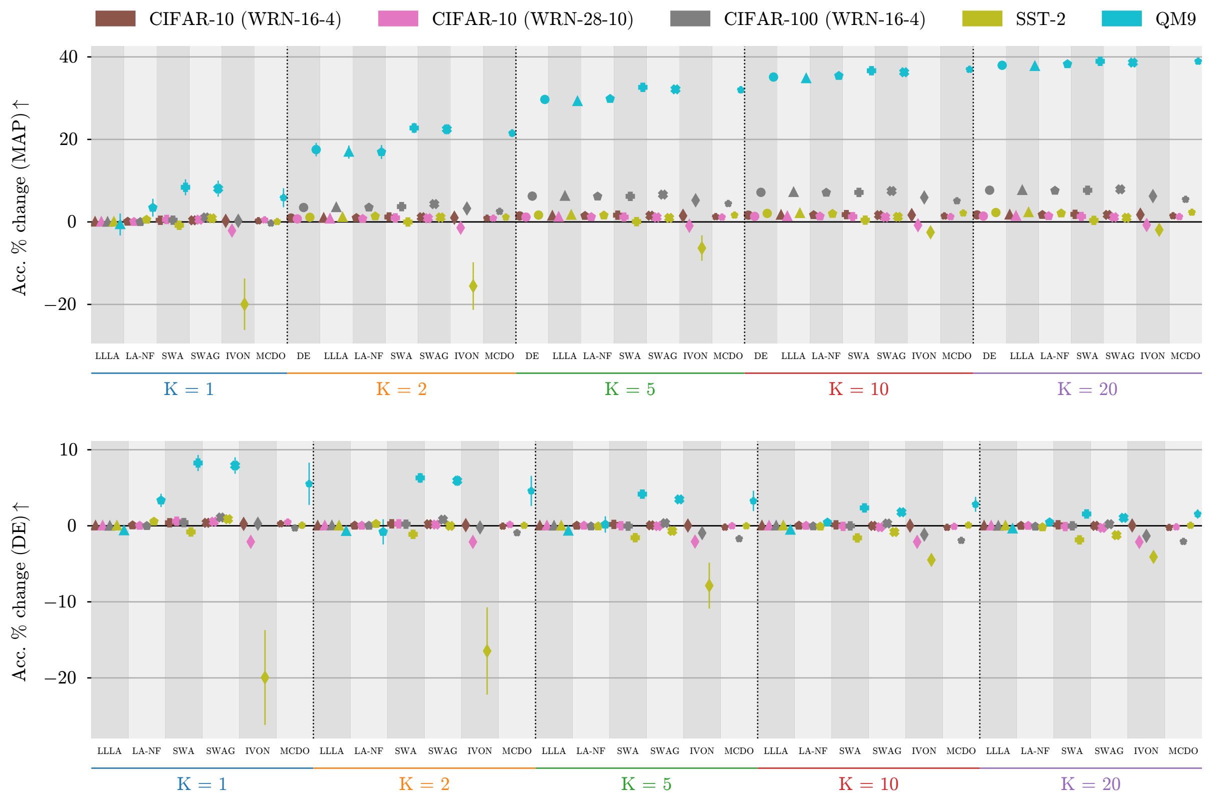

Figure E.2: Accuracy % Change for All Inference Methods and Datasets Excluding IVON on QM9. (top) is versus MAP ($K = 1$) and (bottom) is versus DE with same K.

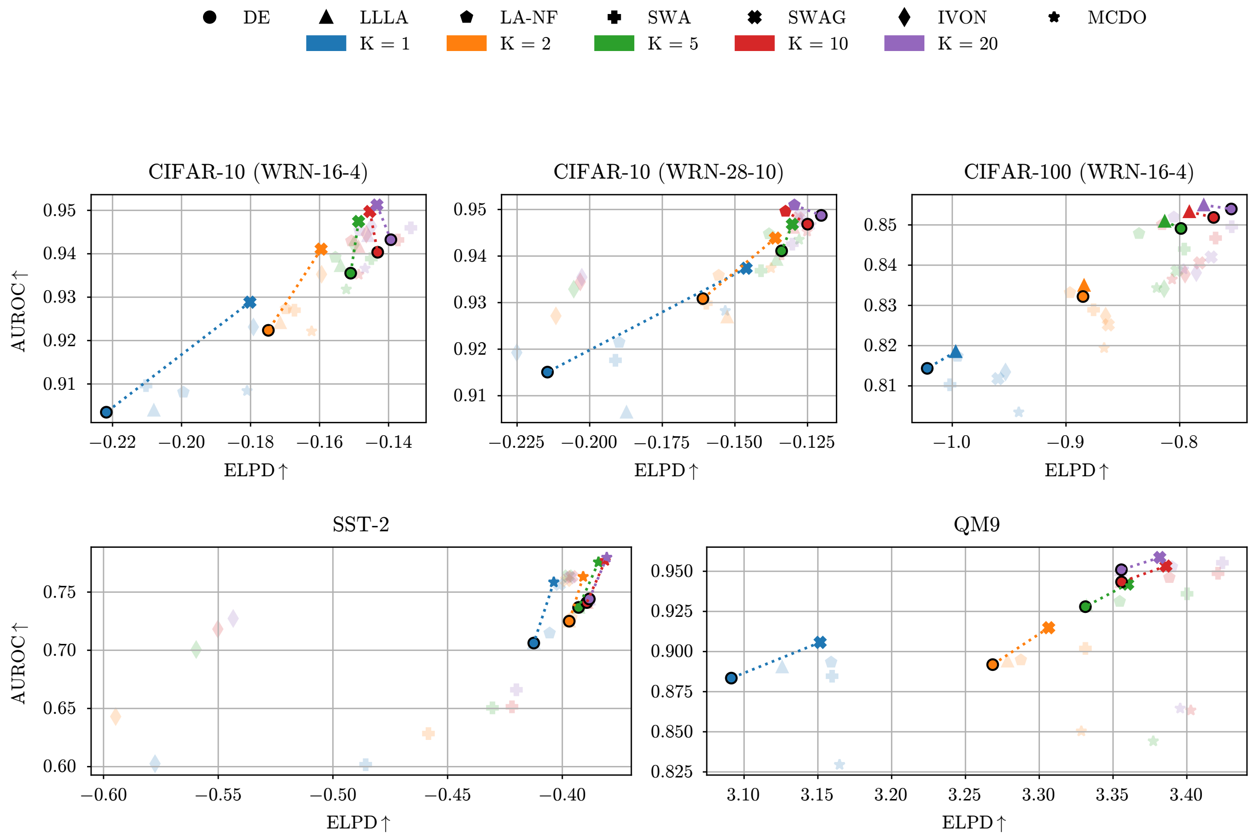

Figure E.3: Out-of-Distribution Performance versus In-Distribution Test Performance for all DEs and DE-BNNs.

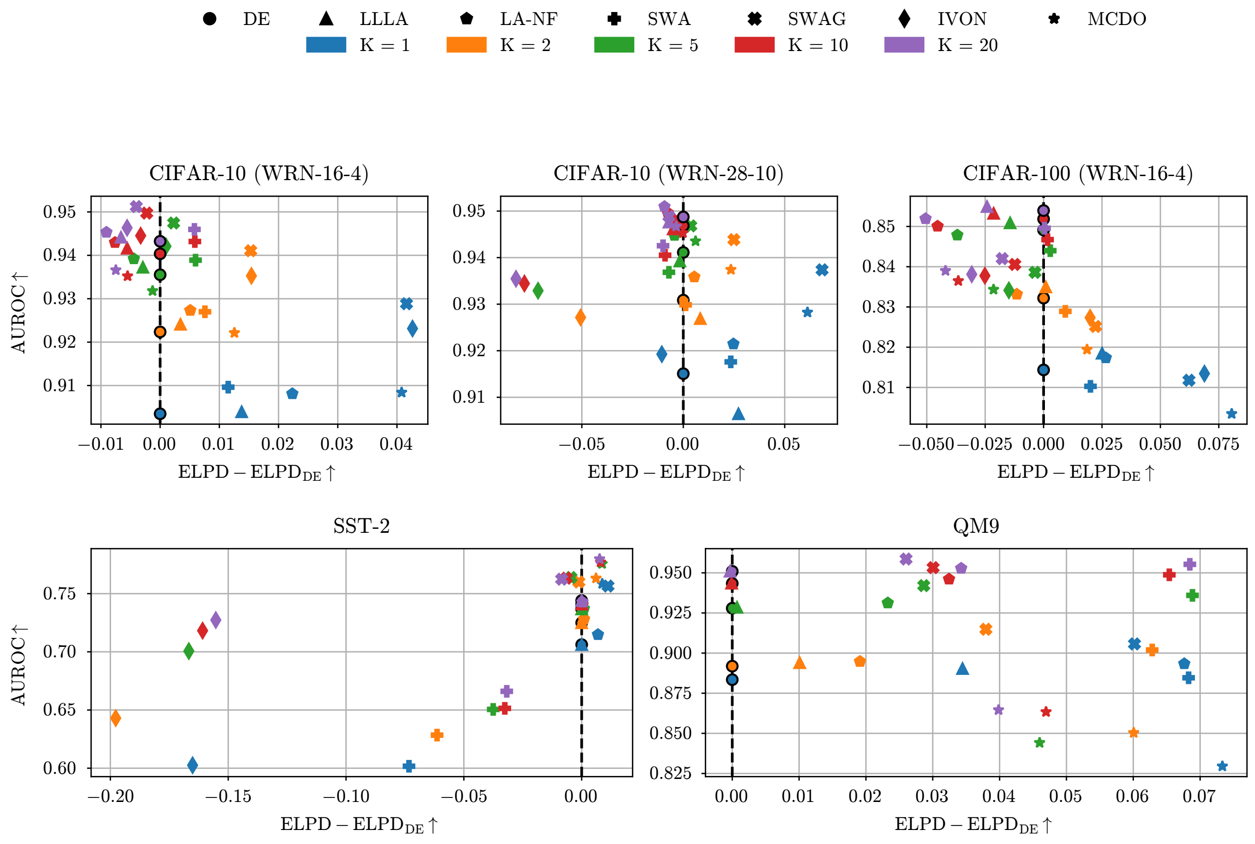

Figure E.4: Out-of-Distribution Performance versus In-Distribution Test Performance for all DEs and DE-BNNs with x-axis being difference in ELPD versus DE.