Make Me a BNN: A Simple Strategy for Estimating Bayesian Uncertainty from Pre-trained Models

Gianni Franchi,[1, *, †] **Olivier Laurent**,[1, 2, *] **Maxence Leguéry**,[1] **Andrei Bursuc**,[3]
Andrea Pilzer[4] & **Angela Yao**[5]
U2IS, ENSTA Paris, Institut Polytechnique de Paris,[1] Université Paris-Saclay,[2]
valeo.ai,[3] NVIDIA,[4] National University of Singapore[5]

Contents

The supplementary material encompasses multiple details and insights complementing the main paper as follows. In Section I, we introduce and clarify the notations used throughout the paper. Theoretical insights are presented in Section A, delving into the theoretical foundations of our methods. Section B evaluates the stability of ABNN, shedding light on its robustness. Section C shifts the focus to the stability of the training procedure, an essential aspect deserving exploration in every post-hoc technique. Section D delves into a sensitivity analysis and ablation study, exploring key components' resilience and performance impact. Section E focuses on the quality of the posterior estimated by ABNN. Additionally, Sections G and F detail the training hyperparameters and showcase additional experiments, providing a comprehensive view of our methodology. Finally, section H outlines the impact of varying the number of finetuning iterations M and the number of sampling instances L on ABNN.

A. Theoretical Analysis

In this section, we develop a mathematical formalism to study the theoretical properties of ABNN.

A.1. Stability of ABNN

Variational inference BNNs [2] are not commonly used in computer vision due to their challenges in scaling properly for deeper high capacity DNNs [4]. In this section, we derive theoretical insights that entail the greater stability of ABNN, arguably also a Variational Inference BNN (VI-BNN). We start with deriving the gradients for a layer in a classic 2-hidden-layer MLP BNN. For the gradient of the loss on the mean of the weights of layer j, we have:

$$\frac{\partial \mathcal{L}_{\text{MAP}}(\boldsymbol{\omega})}{\partial W^{(j)}_{\mu,i,i'}} = \sum_{i''} \frac{\partial \mathcal{L}_{\text{MAP}}(\boldsymbol{\omega})}{\partial \mathbf{h}_{j+1,i''}} \left[W^{(j+1)}_{\mu,i'',i} + \mathcal{E}^{(j+1)}_{i'',i} W^{(j+1)}_{\sigma,i'',i}\right] \frac{a'(\mathbf{h}_{j,i})\mathbf{a}_{j-1,i'}}{\sigma_j}. \quad \text{(A1)}$$

On its standard deviation, we have:

$$\frac{\partial \mathcal{L}_{\text{MAP}}(\boldsymbol{\omega})}{\partial W^{(j)}_{\sigma,i,i'}} = \sum_{i''} \frac{\partial \mathcal{L}_{\text{MAP}}(\boldsymbol{\omega})}{\partial \mathbf{h}_{j+1,i''}} \left[W^{(j+1)}_{\mu,i'',i} + \mathcal{E}^{(j+1)}_{i'',i} W^{(j+1)}_{\sigma,i'',i}\right] \frac{a'(\mathbf{h}_{j,i})\boldsymbol{\epsilon}_{j,i,i'}\mathbf{a}_{j-1,i'}}{\sigma_j}. \quad \text{(A2)}$$

For the gradients of the ABNN parameters, in the case of a 2-hidden-layer MLP BNN, we have:

$$\frac{\partial \mathcal{L}_{\text{MAP}}(\boldsymbol{\omega})}{\partial \gamma^{(j)}_i} = \sum_{i''} \frac{\partial \mathcal{L}_{\text{MAP}}(\boldsymbol{\omega})}{\partial \mathbf{h}_{j+1,i''}} W^{(j+1)}_{i'',i} \frac{\mathbf{h}_j - \mu_j}{\sigma_j}(1+\boldsymbol{\epsilon}_{j,i})a'(\mathbf{u}_{j,i}), \quad \text{(A3)}$$

as well as, on β,

$$\frac{\partial \mathcal{L}_{\text{MAP}}(\boldsymbol{\omega})}{\partial \beta_{j,i}} = \sum_{i''} \frac{\partial \mathcal{L}_{\text{MAP}}(\boldsymbol{\omega})}{\partial \mathbf{h}_{j+1,i''}} W^{(j+1)}_{i'',i} a'(\mathbf{u}_{j,i}). \quad \text{(A4)}$$

We have four random variables: $\frac{\partial \mathcal{L}_{\text{MAP}}(\boldsymbol{\omega})}{\partial \mathbf{h}_{j+1,i''}}$, $\mathcal{E}^{(j+1)}_{i'',i}$ and $\boldsymbol{\epsilon}_{j,i}$ along with $a'(\mathbf{u}_{j,i})$. Let's consider calculating the conditional variance given $\frac{\partial \mathcal{L}_{\text{MAP}}(\boldsymbol{\omega})}{\partial \mathbf{h}_{j+1,i''}}$ for all i''. Assuming that all random variables associated with a single neuron are independent, we have:

$$\text{var}\left(\frac{\partial \mathcal{L}_{\text{MAP}}(\boldsymbol{\omega})}{\partial W^{(j)}_{\mu,i,i'}}\right) = \sum_{i''} \left(\frac{\partial \mathcal{L}_{\text{MAP}}(\boldsymbol{\omega})}{\partial \mathbf{h}_{j+1,i''}} \frac{W^{(j+1)}_{\sigma,i'',i}\mathbf{a}_{j-1,i'}}{\sigma_j}\right)^2 \times \text{var}\left[\mathcal{E}^{(j+1)}_{i'',i} a'(\mathbf{h}_{j,i})\right] \quad \text{(A5)}$$

and

$$\text{var}\left(\frac{\partial \mathcal{L}_{\text{MAP}}(\boldsymbol{\omega})}{\partial W^{(j)}_{\sigma,i,i'}}\right) = \sum_{i''} \left(\frac{\partial \mathcal{L}_{\text{MAP}}(\boldsymbol{\omega})}{\partial \mathbf{h}_{j+1,i''}} \frac{W^{(j+1)}_{\sigma,i'',i}\mathbf{a}_{j-1,i'}}{\sigma_j}\right)^2 \text{var}\left[\mathcal{E}^{(j+1)}_{i'',i} a'(\mathbf{h}_{j,i})\boldsymbol{\epsilon}_{j,i,i'}\right]. \quad \text{(A6)}$$

Using the fact that $\boldsymbol{\epsilon}_{j,i,i'}$ is independent of $\mathcal{E}^{(j+1)}_{i'',i}$ and $a'(\mathbf{h}_{j,i})$ we have that

$$\text{var}\left[\mathcal{E}^{(j+1)}_{i'',i} a'(\mathbf{h}_{j,i})\boldsymbol{\epsilon}_{j,i,i'}\right] = \text{var}\left[\mathcal{E}^{(j+1)}_{i'',i} a'(\mathbf{h}_{j,i})\right] + \mathbb{E}\left(\mathcal{E}^{(j+1)}_{i'',i} a'(\mathbf{h}_{j,i})\right)^2 \quad \text{(A7)}$$

In the case of ABNN, we can express the conditional variance as follows:

$$\text{var}\left(\frac{\partial \mathcal{L}_{\text{MAP}}(\boldsymbol{\omega})}{\partial \gamma^{(j)}_i}\right) = \sum_{i''} \left(\frac{\partial \mathcal{L}_{\text{MAP}}(\boldsymbol{\omega})}{\partial \mathbf{h}_{j+1,i''}} W^{(j+1)}_{i'',i} \frac{\mathbf{h}_j - \mu_{\mathbf{h}_j}}{\sigma_j}\right)^2 \text{var}\left[\boldsymbol{\epsilon}_{j,i} a'(\mathbf{u}_{j,i})\right] \quad \text{(A8)}$$

$$\text{var}\left(\frac{\partial \mathcal{L}_{\text{MAP}}(\boldsymbol{\omega})}{\partial \beta_{j,i}}\right) = \sum_{i''} \left(\frac{\partial \mathcal{L}_{\text{MAP}}(\boldsymbol{\omega})}{\partial \mathbf{h}_{j+1,i''}} W^{(j+1)}_{i'',i}\right)^2 \text{var}\left[a'(\mathbf{u}_{j,i})\right] \quad \text{(A9)}$$

Assuming that $\text{var}\left[\boldsymbol{\epsilon}_{j,i} a'(\mathbf{u}_{j,i})\right] = \text{var}\left[\mathcal{E}^{(j+1)}_{i'',i} a'(\mathbf{h}_{j,i})\right]$, we find that the variances of $\text{var}\left(\frac{\partial \mathcal{L}_{\text{MAP}}(\boldsymbol{\omega})}{\partial W^{(j)}_{\mu,i,i'}}\right)$ and $\text{var}\left(\frac{\partial \mathcal{L}_{\text{MAP}}(\boldsymbol{\omega})}{\partial W^{(j)}_{\sigma,i,i'}}\right)$ are directly proportional to the variance of $\text{var}\left(\frac{\partial \mathcal{L}_{\text{MAP}}(\boldsymbol{\omega})}{\partial \gamma_{j,i}}\right)$.This proportionality is associated with the magnitudes of the weight values, and we assume that they are of similar magnitude. Consequently, the variance of the gradient of the parameters $\beta_{j,i}$ is the smallest among all,

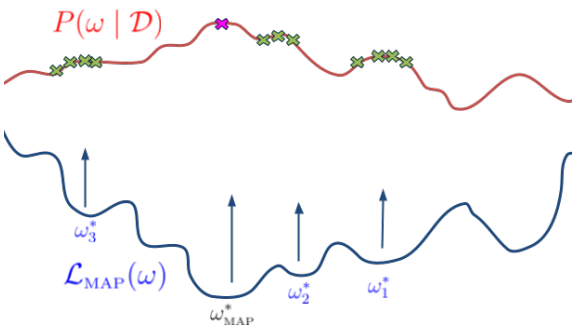

Figure A1. Illustration of the training loss (in blue) and the corresponding posterior distribution (in red). Due to the multi-modal nature of the posterior, training multiple ABNNs with distinct final weights (such as $\boldsymbol{\omega}_1^*$, $\boldsymbol{\omega}_2^*$, and $\boldsymbol{\omega}_3^*$) enables sampling from different modes, enhancing the overall estimation of the posterior.

followed by the variance of the parameters $\gamma_{j,i}$ and $W_{\mu,i,i'}^{(j)}$, which are roughly equivalent, with the highest variance observed for $W_{\sigma,i,i'}^{(j)}$. This property results in more stable backpropagation for ABNN compared to classic VI-BNNs.

A.2. Multi-modes with ABNN

The posterior of the DNN often comprises multiple modes [9, 15], making it non-trivial for an unimodal distribution chosen to represent the BNN's posterior to account for these different modes effectively. One approach to address this issue is to train multiple BNNs, as proposed in the multi-SWAG method by Wilson and Izmailov [15]. However, adapting this strategy to VI-BNNs inherits the instability issue from classic BNNs and may struggle to fit multiple modes accurately.

Our solution, ABNN, also faces a similar challenge, where we need to ensure that the technique doesn't collapse into the same local minima during training. We introduce a small perturbation to the loss function to prevent this collapse, which helps diversify the optimization process. This perturbation involves modifying the class weights within the cross-entropy loss. More precisely, contrary to classic VI-BNN that optimizes the Evidence Lower Bound (ELBO) loss, we propose to maximize the MAP. ABNNs optimize the following loss:

$$\mathcal{L}(\boldsymbol{\omega}) = -\sum_{(\mathbf{x}_i, y_i)\in\mathcal{D}} \alpha(y_i) \log P(y_i \mid \mathbf{x}_i, \boldsymbol{\omega}) - \log P(\boldsymbol{\omega}), \tag{A10}$$

In the standard cross-entropy loss, all classes are given equal weight, typically represented as $\alpha(y_i) = 1$. However, our approach deliberately introduces the *random prior*: random weight adjustments for certain classes, denoted as η_i (such that $\alpha(y_i) = 1 + \eta_i$). This manipulation encourages various ABNNs to specialize as experts in different classes. Consequently, the training loss is formulated as follows:

$$\mathcal{L}(\boldsymbol{\omega}) = \mathcal{L}_{\text{MAP}}(\boldsymbol{\omega}) + \mathcal{E}(\boldsymbol{\omega}) \tag{A11}$$

where

$$\mathcal{E}(\boldsymbol{\omega}) = -\sum_{(\mathbf{x}_i, y_i)\in\mathcal{D}} \eta_i \log P(y_i \mid \mathbf{x}_i, \boldsymbol{\omega}). \tag{A12}$$

Let's denote $\boldsymbol{\omega}^{(0)}$ as the parameter configuration that minimizes $\mathcal{L}_{\text{MAP}}$. Let us suppose for simplicity that the loss function is convex to provide a theoretical grounding to the random prior. After a single step of gradient descent (GD), we have:

$$\boldsymbol{\omega}^{(1)} = \boldsymbol{\omega}^{(0)} - \lambda \nabla \mathcal{L}(\boldsymbol{\omega}^{(0)}), \tag{A13}$$

where $\boldsymbol{\omega}^{(1)}$ represents the parameters after the first optimization step, the superscript denotes the iteration number, and λ is the learning rate.

We can express the updated loss $\mathcal{L}(\boldsymbol{\omega}^{(1)})$ using the GD, as shown in Eq. (A14):

$$\mathcal{L}(\boldsymbol{\omega}^{(1)}) = \mathcal{L}(\boldsymbol{\omega}^{(0)} - \lambda \nabla \mathcal{L}(\boldsymbol{\omega}^{(0)})) \tag{A14}$$

Now, by applying a first-order Taylor expansion to $\mathcal{L}$, we can express the updated loss $\mathcal{L}(\boldsymbol{\omega}^{(1)})$ as a function of the initial loss $\mathcal{L}(\boldsymbol{\omega}^{(0)})$ and the gradient update, as shown in Eq. (A15):

$$\mathcal{L}(\boldsymbol{\omega}^{(1)}) = \mathcal{L}(\boldsymbol{\omega}^{(0)}) - \lambda \nabla \mathcal{L}(\boldsymbol{\omega}^{(0)})^t \nabla \mathcal{L}(\boldsymbol{\omega}^{(0)}) \tag{A15}$$

This equation can be further simplified by noticing that $\nabla \mathcal{L}_{\text{MAP}}(\boldsymbol{\omega}^{(0)}) = 0$:

$$\mathcal{L}(\boldsymbol{\omega}^{(1)}) = \mathcal{L}(\boldsymbol{\omega}^{(0)}) - \lambda \nabla \mathcal{E}(\boldsymbol{\omega}^{(0)})^t \nabla \mathcal{E}(\boldsymbol{\omega}^{(0)}) \tag{A16}$$

Starting with Eq. (A16), we have:

$$\mathcal{L}(\boldsymbol{\omega}^{(1)}) = \mathcal{L}(\boldsymbol{\omega}^{(0)}) - \lambda \sum_{(\mathbf{x}_i, y_i)\in\mathcal{D}} \sum_{(\mathbf{x}_{i'}, y_{i'})\in\mathcal{D}} \eta_i \eta_{i'} \log P(y_{i'} \mid \mathbf{x}_{i'}, \boldsymbol{\omega}^{(0)}) \log P(y_i \mid \mathbf{x}_i, \boldsymbol{\omega}^{(0)}) \tag{A17}$$

Under the assumption that $\lambda \eta_i \eta_{i'}$ is small and that $\log P(y_{i'} \mid \mathbf{x}_{i'}, \boldsymbol{\omega})$ is bounded, we can approximate the loss as follows: $\mathcal{L}(\boldsymbol{\omega}^{(1)}) \simeq \mathcal{L}(\boldsymbol{\omega}^{(0)})$. Consequently, we have $\nabla \mathcal{L}(\boldsymbol{\omega}^{(1)}) \simeq \nabla \mathcal{L}(\boldsymbol{\omega}^{(0)})$, and after another optimization step, $\boldsymbol{\omega}(2)$ is updated as $\boldsymbol{\omega}(2) = \boldsymbol{\omega}^{(1)} - \lambda \nabla \mathcal{L}(\boldsymbol{\omega}^{(1)}) = \boldsymbol{\omega}^{(0)} - 2\lambda \nabla \mathcal{L}(\boldsymbol{\omega}^{(0)})$.

By applying the same technique iteratively, for all t, we can approximate the loss as $\mathcal{L}(\boldsymbol{\omega}^{(t)}) \simeq \mathcal{L}(\boldsymbol{\omega}^{(0)})$. This leads also to the relationship $\boldsymbol{\omega}^{(t)} = \boldsymbol{\omega}^{(0)} - t\lambda \nabla \mathcal{L}(\boldsymbol{\omega}^{(0)})$.

Under the conditions that the loss is convex, the derivatives of the DNN are bounded, and $\lambda \eta_i \eta_{i'}$ is small, we can find minima of $\mathcal{L}$ with similar loss values by introducing

weight diversity. This loss function can be valuable in encouraging each DNN to escape from the global minima, particularly in convex cases. In non-convex cases, standard SGD may already help escape from local minima, but this additional loss may offer extra assistance in avoiding the same local minima. In Figure A1, we present a visualization depicting the training loss alongside the posterior distribution. This Figure highlights the importance of training multiple ABNNs with different optimal solutions to improve the quality in estimating the posterior distribution.

A.3. Discussion on the Bayesian Neural Network Nature of ABNN

The Law of the Unconscious Statistician (LOTUS) [5] is a theorem in probability theory that offers a method for computing the expected value of a function of a random variable. Hence, for a continuous random variable X with a probability density function $f_X(x)$, the expected value of a function $Y = g(X)$ is expressed as:

$$E_Y(Y) = E_X[g(X)] \tag{A18}$$

In our scenario, let U_j denote the random variable associated with $\mathbf{u}_j$, and $W^{(j)}$ represent the random variable $W^{(j)}$ (we use the same letter for simplification). Thus, we have:

$$E_{U_j}(U_j) = \frac{\mathbf{h}_j - \mu_{\mathbf{h}_j}}{\sqrt{\sigma^2_{\mathbf{h}_j} + \epsilon}}\gamma_j + \beta_j \tag{A19}$$

For simplification, we set β_j=0, which leads to

$$\text{var}_{U_j}(U_j) = \left(\frac{\mathbf{h}_j - \mu_{\mathbf{h}_j}}{\sqrt{\sigma^2_{\mathbf{h}_j} + \epsilon}}\gamma_j\right)^2 \tag{A20}$$

Here, the parameters γ_j and β_j are optimized to obtain the best Bayesian Neural Network (BNN)

B. Experiment on the variance of the gradient

To validate our hypothesis that ABNN is more stable than VI-BNNS, as detailed in section A.1, we analyze the variances of the gradients of classic DNNs, VI-BNNs, and ABNNs. Table A1 reveals that the gradient variance of BNNs are significantly greater than that of DNNs, aligning with the inherent challenges in training BNNs. Notably, ABNN exhibits a considerably lower gradient, stemming from only the weights of BNL are trained. Both empirical observations and theoretical considerations affirm the superior stability of ABNN on this side. Additionally, Table A1 includes results for a VI-BNN, which, as discussed in this section, exhibits suboptimal performance. Noteworthy other works [4, 7] also express concerns regarding the stability of VI-BNNs.

Method	**variance** (10^{-4})
ResNet50	1.43
ResNet50 BNN	2.54
ResNet50 ABNN	$3.20{\cdot}10^{-2}$
Wide-ResNet	1.37
Wide-ResNet BNN	2.53
Wide-ResNet ABNN	$9.91{\cdot}10^{-2}$

Table A1. **Variance of gradients on relevant parameters**
Single model: every weight (no bias)
BNN: means of the weight samplers
ABNN: parameters linked to the BNL layers

	Acc	**ECE**	**AUPR**	**AUC**	**FPR95**
Single model	0.356	0.0002	1.036	1.445	2.740
one ω_{MAP} + Multiple ABNN	0.066	0.0009	0.130	0.224	0.658
Multiple ω_{MAP} + ABNN	0.324	0.0011	1.131	1.570	3.202

Table A2. **Standard Deviation (SD) Comparison of ABNN and DNN.** The first row presents the SD of a single DNN, while the second row depicts the SD of ABNN starting from a single checkpoint. The last row quantifies the SD of ABNN when trained from different checkpoints. All training scenarios use the optimal hyperparameters for ABNN on a ResNet-50 architecture on the CIFAR-10 dataset.

C. Discussion on stability of the training of ABNN

ABNN being a post-hoc technique, it is imperative to ensure that it does not introduce instability to DNNs, especially in the critical domain of uncertainty quantification. We train multiple single models based on a ResNet-50 architecture on CIFAR-10 to verify this point, calculating the standard deviation of the different metrics. Additionally, we derive several ABNNs starting from these checkpoints and assess the variance. Finally, we apply our technique to train an ABNN for each checkpoint of the single models. Table A2 demonstrates that our approach minimally increases the variance, confirming that it does not introduce instability to the uncertainty quantification process.

D. Ablation study of ABNN and Sensitivity analysis

In Table A3, we conduct a study to inspect the impact of adding or removing two characteristics from our method. First, we investigate whether the addition of the random prior (linked to $\mathcal{E}$ term in Eq. (A12)), which introduces disturbance to the loss, improves the performance of ABNN. We test the Random Prior (RP) to check whether $\mathcal{E}$ contributes positively. Notably, when training a single RP model (when **MM** is ✗), it appears to degrade performance as it corrupts the cross-entropy. Conversely, in the case of training multiple ABNNs (**MM** is ✓), RP seems to improve uncertainty quantification metrics. The second aspect is the training of multiple modes: Table A3 shows that incorporating multiple modes (**MM** is ✓) improves the quality of the uncertainty quantification, in particular for OOD detection.

We analyze the performance variations in Tables A4 and A5 by modifying the learning rate during the fine-tuning phase. It's essential to highlight that this hyperparameter is the only one of ABNN. We fine-tune a single model with various learning rates for this evaluation after adapting it to ABNN. Remarkably, the learning rate appears non-critical, as the performances on CIFAR-10 and CIFAR-100 exhibit minimal variations, around one percent.

E. Discussion on diversity of ABNN

We trained a Deep Ensembles of ResNet-50 architecture for 200 epochs, specifically three networks. As demonstrated in [6], Deep Ensembles strikes a good balance between accuracy and diversity, enabling effective uncertainty quantification. Simultaneously, we train three ABNNs, initializing them from the same checkpoint. It's important to highlight that all ABNN parameters were trained for this experiment. To visually compare the results, akin to [6] (Figure 2.c), we conduct a t-SNE analysis on the latent space of all the checkpoints after each epoch. Initially, ABNN shows limited diversity due to having only one checkpoint. However, as training progresses, some diversity emerges, though less than observed in Deep Ensembles. Table A6 illustrates that ABNN exhibits lower mutual information than BNN on both in-distribution (IDs) and out-of-distribution (OOD) samples. Yet, interestingly, ABNN achieves a superior mutual information ratio on OODs/IDs. It's worth noting that mutual information serves as a metric for measuring diversity and can also quantify epistemic uncertainty. Mutual information is defined for a finite set $\{\boldsymbol{\omega}_1, \dots \boldsymbol{\omega}_M\}$ of M weight configurations sampled from the posterior distribution as :

$$\underbrace{I\left(P(\boldsymbol{\omega} \mid \mathcal{D})\right)}_{\text{Epistemic uncertainty}} = \underbrace{\mathcal{H}\left(\frac{1}{M}\sum_{m=1}^{M} P(y \mid \mathbf{x}, \boldsymbol{\omega}_m)\right)}_{\text{Total uncertainty}} - \underbrace{\frac{1}{M}\sum_{m=1}^{M} \mathcal{H}(P(y \mid \mathbf{x}, \boldsymbol{\omega}_m))}_{\text{Aleatoric uncertainty}}. \quad \text{(A21)}$$

with $\mathcal{H}(\cdot)$ the entropy is defined by :

$$\mathcal{H}(P(y \mid \mathbf{x}, \boldsymbol{\omega})) = -\sum_{y} P(y_i \mid \mathbf{x}_i, \boldsymbol{\omega}) \log P(y_i \mid \mathbf{x}_i, \boldsymbol{\omega}) \quad \text{(A22)}$$

The good mutual information ratio highlights ABNN's superior ability to detect out-of-distribution samples compared to BNN. Moreover, Figure **??** illustrates that benefiting from multiple training instances, ABNN can effectively model multi-modes—a capability beyond the reach of classical BNNs. This is particularly valuable given the inherent multi-modal nature of the posterior.

F. Extra Experiments

F.1. ViT transfer learning

We also show that ABNN can be used in contexts of transfer learning. Table A7 presents the comparative performance of ViT B-16 pre-trained on ImageNet-21k and fine-tuned with and without a mono-modal ABNN on CIFAR-100 in 10,000 steps. Despite the mono-modality of ABNN in this experiment, we show that the corresponding ViT outperforms the classic fine-tuning in calibration and AUPR.

F.2. Extra baselines

To benchmark our method against other posthoc techniques, we we implement a variant of Test-Time Augmentation [1, 10, 11], incorporating random Gaussian noise with a specified standard deviation of 0.08. The objective is to introduce diversity and ensemble different predictions by leveraging the added noise. Like [8], we experimented with adding noise to the latent space, representing the scenario where ABNN is not trained. We tested various levels of standard deviation, and the corresponding results are summarized in Table A8. Notably, the untrained ABNN does not perform effectively, underscoring the significance of a brief finetuning phase. Additionally, we train a VI-BNN, a non-posthoc technique, to understand the performance of a traditional BNN. It's noteworthy that VI-BNNs proved challenging to train and demonstrated subpar performance.

		RP	MM	Acc ↑	ECE ↓	AUPR ↑	AUC ↑	FPR95 ↓
CIFAR-10	ResNet-50	✗	✗	94.7	1.0	96.8	94.1	17.3
		✓	✗	95.2	**0.9**	96.7	94.4	15.3
		✗	✓	95.3	1.34	**97.1**	**94.8**	15.7
		✓	✓	**95.4**	**0.9**	97.0	94.7	**15.1**
	WideResNet	✗	✗	94.4	**1.34**	96.4	93.7	18.2
		✓	✗	92.8	2.2	97.5	95.3	14.8
		✗	✓	**94.9**	1.38	97.3	95.1	15.7
		✓	✓	93.7	1.8	**98.5**	**96.9**	**12.6**
CIFAR-100	ResNet-50	✗	✗	78.8	5.7	89.3	80.8	50.4
		✓	✗	78.7	**5.5**	89.4	81.0	50.1
		✗	✓	78.3	5.8	89.6	**81.6**	**48.2**
		✓	✓	78.8	5.6	**89.7**	**81.6**	49.0
	WideResNet	✗	✗	**80.5**	5.6	84.0	72.6	62.2
		✓	✗	79.6	**5.5**	84.7	74.9	55.8
		✗	✓	80.4	**5.5**	85.0	75.0	57.7
		✓	✓	78.2	5.9	**87.8**	**79.7**	**49.0**

Table A3. **Performance comparison (averaged over five runs) on CIFAR-10/100 using ResNet-50 and WideResNet28×10. RP** is Random Prior, and **MM** is multi-mode. For OOD detection, we use the SVHN dataset.

Coef. LR	Acc ↑	ECE ↓	AUPR ↑	AUC ↑	FPR95 ↓
$\times 10^{-1}$	95.5	0.9	96.2	93.0	20.7
$\times 2 \cdot 10^{-1}$	95.4	0.9	96.5	93.8	18.2
$\times 5 \cdot 10^{-1}$	95.4	1.0	96.3	93.3	19.9
$\times 1$	95.4	0.9	97.0	94.7	15.1
$\times 2$	95.3	1.1	95.7	92.3	22.0
$\times 5$	94.6	1.5	96.1	93.1	19.6
$\times 10$	93.9	1.1	96.7	94.2	19.2

Table A4. **Sensitivity Analysis of the Learning Rate on CIFAR-10.** We conducted training for ABNN using a learning rate set at 0.0057 multiplied by the **Coef LR**.

Coef. LR	Acc ↑	ECE ↓	AUPR ↑	AUC ↑	FPR95 ↓
$\times 10^{-1}$	79.0	5.4	89.0	80.3	51.5
$\times 2 \cdot 10^{-1}$	78.9	5.5	88.6	79.9	52.5
$\times 5 \cdot 10^{-1}$	78.9	5.6	88.9	80.0	52.5
$\times 1$	78.9	5.5	89.4	81.0	50.1
$\times 2$	78.8	5.7	88.6	79.8	52.6
$\times 15$	79.0	5.5	88.8	80.1	52.0
$\times 10$	78.8	5.7	88.8	80.2	51.4

Table A5. **Sensitivity Analysis of the Learning Rate on CIFAR-100.** We conducted training for ABNN using a learning rate set at 0.00139 multiplied by the **Coef. LR**.

G. Training hyperparameters

Table A9 provides a detailed overview of all the hyperparameters employed throughout our study. We use SGD in conjunction with a multistep learning-rate scheduler for image classification tasks, adjusting the rate by multiplying it by γ-lr at each milestone. It's important to note that, for stability reasons, BatchEnsemble based on ResNet-50 employed a lower learning rate of 0.08, deviating from the default 0.1. Our "Medium" data augmentation strategy encompasses a blend of Mixup [17] and Cutmix [16], with a switch probability of 0.5. Additionally, `timm`'s augmentation classes [13] were incorporated with coefficients of 0.5 and 0.2. RandAugment [3] with parameters $m = 9$, $n = 2$, and $mstd = 1$, along with label-smoothing [12] of intensity

	ID	OOD	OOD/ID
ABNN	1.39e-4	5.22e-4	**3.76**
BNN	0.139	0.179	1.28

Table A6. **Evaluation of the average Mutual Information** on the test set and the OOD set of the different sample from an ABNN or a BNN.

	Acc	ECE	AUPR	AUC	FPR95
Single model	**92.0**	4.4	96.6	**92.7**	**28.1**
ABNN	91.9	**1.4**	**96.9**	92.5	33.9

Table A7. **Transfer learning of a ViT pre-trained on ImageNet-21k on CIFAR-100.** The first line is the classic pre-trained model fine-tuned with ABNN, and the second is fine-tuned with the classic layer normalization.

		Acc ↑	ECE ↓	AUPR ↑	AUC ↑	FPR95 ↓
CIFAR-10	Single model	**95.53**	**0.83**	96.52	93.70	18.43
	VI BNN	75.66	5.40	80.60	69.53	66.62
	Test-Time Augmentation	89.95	2.49	93.78	89.95	25.12
	Noise on latent space std=0.01	95.51	0.82	96.51	93.68	18.56
	Noise on latent space std=0.1	95.46	0.90	95.88	92.64	20.94
	Noise on latent space std=1	18.66	31.58	71.89	50.86	93.27
	ABNN	95.43	0.85	**97.03**	**94.73**	**15.11**
CIFAR-100	Single model	**79.05**	**5.34**	88.72	79.96	52.04
	VI BNN	41.17	8.97	78.44	61.82	86.15
	Test-Time Augmentation	72.65	7.67	86.64	76.89	55.52
	Noise on latent space std=0.01	79.03	5.39	88.68	79.85	52.54
	Noise on latent space std=0.1	77.49	6.36	86.53	75.19	64.42
	Noise on latent space std=1	1.03	11.91	74.08	50.92	95.74
	ABNN	78.94	5.47	**89.36**	**81.04**	**50.12**

Table A8. **Performance comparison of different Post-hoc and BNN uncertainty quantification baselines on CIFAR-10/100 using ResNet-50.**

0.1, were also applied.

In the case of ImageNet, we follow the A3 procedure outlined in [14] for all models. It's worth mentioning that training according to the exact A3 procedure was not consistently feasible; please refer to the specific subsections for additional details.

We highlight that, to enhance training stability and fasten the training, we introduc a hyperparameter α in the BNL layer. This transforms the layer as follows:

$$\mathbf{BNL}(\mathbf{h}_j) = \frac{\mathbf{h}_j - \hat{\mu}_j}{\hat{\sigma}_j} \times \gamma_j(1 + \boldsymbol{\epsilon}_j\alpha) + \beta_j. \tag{A23}$$

(A24)

The hyperparameter α is typically set to 0.01, except in the case of ViT, where specific considerations may apply.

H. Discussion on the influence of the parameter L and M

In table A11, we observe the evolution of parameter M for ABNN. It is noticeable that this parameter increases consistently with the number of fine-tuned DNNs (M). This indicates that by expanding the size of the ensemble, performance gains can potentially be achieved. As for parameter L, we observe minimal variations. This is likely because training with Gaussian noise (present in BNL) might turn the DNN relatively robust to these perturbations. Therefore, we advocate against using a large value for L.

I. Notations

We summarize the main notations used in the paper in Table A12.

Dataset	**Networks**	Epochs	Batch size	start lr	Momentum	Weight decay	γ-lr	Milestones	Data augmentations
C10	R50	200	128	0.1	0.9	$5{\cdot}10^{-4}$	0.2	60, 120, 160	HFlip
C10	WR28-10	200	128	0.1	0.9	$5{\cdot}10^{-4}$	0.2	60, 120, 160	HFlip
C100	R50	200	128	0.1	0.9	$5{\cdot}10^{-4}$	0.2	60, 120, 160	HFlip
C100	WR28-10	200	128	0.1	0.9	$5{\cdot}10^{-4}$	0.2	60, 120, 160	Medium

Table A9. **Hyperparameters for the image classification experiments.** HFlip denotes the classical horizontal flip.

Dataset	**Networks**	Epochs	Batch size	start lr	Alpha	RandomPrior	Momentum	Weight decay	γ-lr	Milestones	Data augmentations
C10	R50	5	128	0.01	0.1	7	0.9	10^{-4}	/	/	HFlip
C10	WR28-10	5	128	0.01	0.1	7	0.9	10^{-4}	/	/	HFlip
C100	R50	5	128	0.01	0.1	7	0.9	10^{-4}	/	/	HFlip
C100	WR28-10	5	128	0.01	0.1	7	0.9	10^{-4}	/	/	HFlip
Imagenet	R50	1	128	0.0044	0,01	8	0.9	$5{\cdot}10^{-4}$	/	/	HFlip
Imagenet	ViT	0.25	128	$7.33{\cdot}10^{-6}$	0.0004	2	0.9	$7 \cdot 10^{-6}$	$5 \cdot 10^{-4}$	Constant	HFlip
StreetHazards	DeepLabv3+	10	4	0.01	0.01	/	0.9	10^{-4}	0.9	Polynomial	HFlip, RandomCrop, ColorJitter, RandomScale
BDD-Anomaly	DeepLabv3+	10	4	0.01	0.01	/	0.9	10^{-4}	0.9	/	HFlip, RandomCrop, ColorJitter, RandomScale
MUAD	DeepLabv3+	6	4	0.044	0.01	/	0.9	10^{-4}	0.9	/	HFlip, RandomCrop, ColorJitter, RandomScale

Table A10. **Hyperparameters for the image classification experiments with ABNN.** HFlip denotes the classical horizontal flip. Random prior has been used.

	M	**Acc** $\uparrow$	**ECE** $\downarrow$	**AUPR** $\uparrow$	**AUC** $\uparrow$	**FPR95** $\downarrow$
C.-10	3	94.74	1.19	96.96	94.64	17.05
	4	95.02	0.86	96.95	94.78	16.08
	5	94.97	0.73	97.06	94.84	16.01

Table A11. **Sensitivity Analysis** on the number of samples M and L on CIFAR-10 using ResNet-50.

Table A12. **Summary of the main notations of the paper.**

Notations	**Meaning**
$\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^N$	The set of N data samples and the corresponding labels
j	The index of the current layer
$\boldsymbol{\omega}$	The set of all the weights of the DNN
$\boldsymbol{\omega}_m \sim P(\boldsymbol{\omega}\|\mathcal{D})$	The m-th sample from the concatenation of weights of the posterior of the DNN.
M	The number of networks in an ensemble
$\boldsymbol{\omega}^{(t)}$	The concatenation of all the weights of the DNN after t steps of optimization
$\mathbf{h}_j$	The pre-activation feature map and output of layer $(j-1)$ & input of layer j before normalization
$\mathbf{u}_j$	The pre-activation feature map and output of layer $(j-1)$ & input of layer j before normalization
γ_j β_j	The parameters of the batch, instance, or layer normalization of layer j
μ_j σ_j	The empirical mean and variance used by the batch, instance, or layer normalization of layer j
$\mathbf{a}_j$	The feature map and output of layer j, $\mathbf{a}_j = a(\mathbf{u}_j)$
$a(\cdot)$	The activation function
$W^{(j)}$	The weights of the j-th layer in a Multi-Layer Perceptron (MLP).
$W_\mu^{(j)}$	The mean weights of the j-th layer in a BNN MLP
$W_\sigma^{(j)}$	The standard variation weights of the j-th layer in a BNN MLP
$\epsilon^{(j)} \sim \mathcal{N}(\mathbf{0}, \mathbb{1})$	A vector sampled from a standard normal distribution at layer j
ϵ	The concatenation of all the $j-$ the $\epsilon^{(j)}$ of each layer j
ϵ_l	The l-th sample of ϵ
$\mathcal{H}$	The entropy function

References

[1] Arsenii Ashukha, Alexander Lyzhov, Dmitry Molchanov, and Dmitry Vetrov. Pitfalls of in-domain uncertainty estimation and ensembling in deep learning. In *ICLR*, 2019. 5

[2] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *ICML*, 2015. 2

[3] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *CVPR*, 2020. 6

[4] Michael Dusenberry, Ghassen Jerfel, Yeming Wen, Yian Ma, Jasper Snoek, Katherine Heller, Balaji Lakshminarayanan, and Dustin Tran. Efficient and scalable bayesian neural nets with rank-1 factors. In *ICML*, 2020. 2, 4

[5] William Feller. *An introduction to probability theory and its applications, Volume 2*. John Wiley & Sons, 1991. 4

[6] Stanislav Fort, Huiyi Hu, and Balaji Lakshminarayanan. Deep ensembles: A loss landscape perspective. *arXiv preprint arXiv:1912.02757*, 2019. 5

[7] Jiri Hron, Roman Novak, Jeffrey Pennington, and Jascha Sohl-Dickstein. Wide bayesian neural networks have a simple weight posterior: theory and accelerated sampling. In *ICML*, 2022. 4

[8] John Lambert, Ozan Sener, and Silvio Savarese. Deep learning under privileged information using heteroscedastic dropout. In *CVPR*, pages 8886–8895, 2018. 5

[9] Olivier Laurent, Emanuel Aldea, and Gianni Franchi. A symmetry-aware exploration of bayesian neural network posteriors. In *ICLR*, 2024. 3

[10] Divya Shanmugam, Davis Blalock, Guha Balakrishnan, and John Guttag. Better aggregation in test-time augmentation. In *ICCV*, 2021. 5

[11] Jongwook Son and Seokho Kang. Efficient improvement of classification accuracy via selective test-time augmentation. *IS*, 2023. 5

[12] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016. 6

[13] Ross Wightman. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019. 6

[14] Ross Wightman, Hugo Touvron, and Herve Jegou. Resnet strikes back: An improved training procedure in timm. In *NeurIPSW*, 2021. 7

[15] Andrew G Wilson and Pavel Izmailov. Bayesian deep learning and a probabilistic perspective of generalization. *NeurIPS*, 2020. 3

[16] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *CVPR*, 2019. 6

[17] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR*, 2018. 6