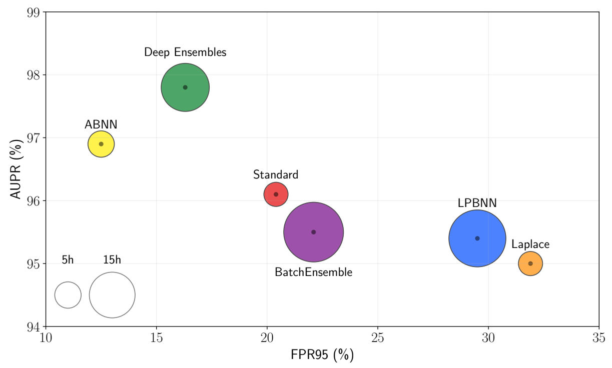
This CVPR paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

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]



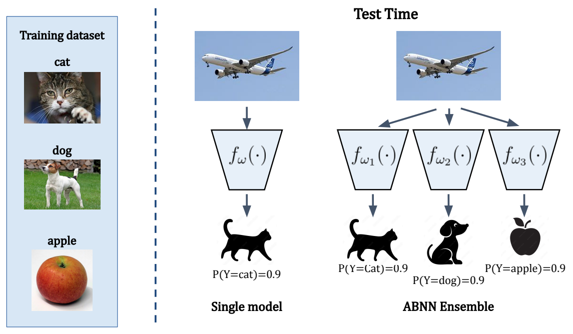


Figure 1. **Benefits of ABNNs. (left)** Trade-off between computational cost (training time) and performance (FPR95 score, lower the better) for various uncertainty quantification techniques on CIFAR-10 with WideResNet-28x10 and ensembles of size 3. The circle area is proportional to the training time. **(right)** Benefits of using ABNN ensembling at test time. Notably, given an out-of-distribution sample, a simple DNN may make high-confidence incorrect predictions, whereas ABNN produces uncertain decisions through its diverse predictions.

Abstract

Deep Neural Networks (DNNs) are powerful tools for various computer vision tasks, yet they often struggle with reliable uncertainty quantification — a critical requirement for real-world applications. Bayesian Neural Networks (BNN) are equipped for uncertainty estimation but cannot scale to large DNNs where they are highly unstable to train. To address this challenge, we introduce the Adaptable Bayesian Neural Network (ABNN), a simple and scalable strategy to seamlessly transform DNNs into BNNs in a post-hoc manner with minimal computational and training overheads. ABNN preserves the main predictive properties of DNNs while enhancing their uncertainty quantification abilities through simple BNN adaptation layers (attached to normalization layers) and a few fine-tuning steps on pre-trained models. We conduct extensive experiments across multiple datasets for image classification and semantic segmentation tasks, and our results demonstrate that ABNN achieves state-of-the-art performance without the computational budget typically associated with ensemble methods.

[*]equal contribution, [†] gianni.franchi@ensta-paris.fr

1. Introduction

Deep Neural Networks (DNNs) have emerged as powerful tools with a profound impact on various perception tasks, such as image classification [18, 50], object detection [32, 70], natural language processing [17, 69, 76], etc. With this progress, there is growing excitement and expectation about the potential applications of DNNs across industries. To meet this end, there is a critical need to address a fundamental challenge: improving DNN reliability by quantifying the inherent uncertainty in their predictions [36, 37, 83]. Deploying DNNs in real-world applications, particularly in safety-critical domains such as autonomous driving, medical diagnoses, industrial visual inspection, etc., requires a comprehensive understanding of their limitations and vulnerabilities beyond their raw predictive accuracy, often considered a main performance metric. By quantifying the uncertainty within these models with millions of parameters and non-trivial inner-working [1, 94] and failure modes [33, 66] in front of the many different long-tail scenarios [7, 56], we can make informed decisions about when and how to rely on their predictions.

In deep learning, uncertainty estimation has been traditionally addressed under Bayesian approaches drawing inspiration from findings in Bayesian Neural Networks (BNNs) [5, 58, 64, 86] that stand on solid theoretical grounds and properties [44, 62, 90]. BNNs estimate the posterior distribution of the model parameters given the training dataset. At runtime, ensembles can be sampled from this distribution, and their predictions can be averaged for reliable decisions. BNNs promise improved predictions and reliable uncertainty estimates with intuitive decomposition of the uncertainty sources [16, 41]. However, although they are easy to formulate, BNNs are notoriously difficult to train over large DNNs [20, 67], in particular for complex computer vision tasks, due to training instability and computational inefficiency as they are typically trained through variational inference [45]. This limitation of BNNs has inspired two major lines of research toward scalable uncertainty estimation: ensembles and last-layer uncertainty approaches.

Deep Ensembles [51] emerge as a simple and highly effective alternative to BNNs for uncertainty estimation on large DNNs. Deep Ensembles are trivial to train by essentially instantiating the same training procedure over different weight initializations of the same network and have been shown to preserve many of the properties of BNNs in terms of predictions diversity [22, 90]. This is a beneficial property for out-of-distribution (OOD) generalization [63]. However, their high computational cost (during both training and inference) makes them inapplicable to many practical applications with computational constraints. In the last few years, multiple computationally efficient alternatives have emerged aiming to reduce training cost during training and/or inference [13, 23, 25, 26, 30, 59, 87]. However, these methods propose specific network architectures and non-trivial trade-offs in computational cost, accuracy, and predictive uncertainty quality.

Last-layer uncertainty approaches aim for BNNs with fewer stochastic weights to produce ensembles or uncertainty estimates [6, 9, 12, 48, 60]. These methods leverage popular DNN architectures [31] to which they attach a stochastic layer and train all parameters for a full training cycle. While training stability is improved compared to standard BNNs, joint optimization of deterministic and stochastic parameters requires careful tuning. Daxberger et al. [12] propose training the last layer separately in a *post-hoc* manner effectively leveraging Laplace approximation for optimization [58, 72, 81]. Decoupling the optimization of the encoder from the uncertainty layer enables the use of the typical training recipes for the encoder or simply leveraging off-the-shelf pre-trained networks. The limitation of last-layer methods is related to the access to only high-level features for producing uncertainty estimates, whereas signals of distribution shift or small anomaly patterns (e.g., in semantic segmentation) can be found mostly in low-level features earlier in the network. Indeed, strong uncertainty estimation methods leverage information from multiple layers of the networks [13, 25, 55].

In this work, we aim for scalable and effective uncertainty estimation without sophisticated optimization schemes and potential training instability and without compromising predictive performance. We propose a post-hoc strategy that starts from a pre-trained DNN and transforms it into a BNN with a simple plug-in module attached to the normalization layers and only a few epochs of fine-tuning. We show that this strategy, dubbed Adaptable-BNN (ABNN), can estimate the posterior distribution around the local minimum of the pre-trained model in a resource-efficient manner while still achieving competitive uncertainty estimates with diversity. Furthermore, ABNN allows for sequential training of multiple BNNs starting from the same checkpoint, thus modeling various modes within the true posterior distribution.

To summarize, our contributions are: **(1)** We propose ABNN, a simple strategy to transform a pre-trained DNN into a BNN with uncertainty estimation capabilities. ABNN is computationally efficient and compatible with multiple DNN architectures (ConvNets: ResNet-50, WideResnet28-10; ViTs), provided they are equipped with normalization layers. **(2)** We observe that the variance of the gradient for ABNN's parameters is lower compared to that of a classic BNN, resulting in a more stable backpropagation. **(3)** Extensive experiments validate that ABNN, although simple and computationally frugal, achieves competitive performance in terms of accuracy and uncertainty estimation over multiple datasets and tasks: image classification (CIFAR-10, CIFAR-100 [49], ImageNet [15]) and semantic segmentation (StreetHazards, BDD-Anomaly [35], MUAD [24]) in both in- and out-of-distribution settings. We train our models using TorchUncertainty *(https://github.com/ENSTA-U2IS-AI/torch-uncertainty)* and make all our configuration files and supplementary scripts available on GitHub.

2. Related work

Epistemic uncertainty and Bayesian posterior. Tackling epistemic uncertainty estimation [39] – the uncertainty on the model itself – is essential to improve the reliability of DNNs [41]. However, obtaining satisfying approximations of this uncertainty remains a challenge as it requires a scalable estimation of the extremely high dimensional distribution of the weights, the *posterior*. Our work presents ABNN, a significantly more scalable method compared to the BNNs [28] that predominantly shape the landscape of epistemic uncertainty estimations [27].

Bayesian Neural Networks and Ensembles. BNNs [82] formulate probabilistic predictions by both introducing explicit and controllable prior knowledge on the network weights [46, 64] and estimating the posterior. While mathematically formulating the posterior distribution is possible [90], its computation for modern models is intractable. This need for scalability leads to approximation techniques that include variational inference BNNs [4, 5], which fit simpler distributions to the posterior (diagonal Gaussian for the former). Many other approximation methods have been proposed, like efficient probabilistic back-propagation [38], Monte-Carlo Dropout [26] that model the posterior as a mixture of Diracs and Laplace methods that estimate the posterior thanks to the local curvature of the loss [58, 72, 74, 81]. However, *deep ensembles* [29, 51], the most successful solution is simpler. It consists of simply averaging the predictions of several independently trained models. This method is powerful as it improves the reliability and the quality of the predictions, albeit costly in training and inference. Many approximate methods stepped into the breach and proposed to reduce the number of parameters, the training time, or the number of forward passes [23, 25, 30, 53, 87, 91]. The proposed method is in this line of scalability and computational efficiency.

Post-hoc uncertainty quantification. In today's context of computer vision, with ever-increasing datasets [47, 75], model sizes [14], and inference constraints, post-hoc methods could be a solution to benefit from both the power of foundation models [42, 73, 96] and uncertainty quantification. The Laplace method reigns among these post-hoc uncertainty quantification methods, especially with its last-layer approximations [72] that make them even more scalable. However, even with the coarse approximation of the posterior achieved by Kronecker-factored Laplace [74], it is not always very efficient with modern datasets. We propose a straightforward and effective method that works with pre-trained models.

3. Background

We start by introducing our formalism and a brief overview of the Bayesian posterior and BNNs for uncertainty quantification.

3.1. Preliminaries

Notations. Let us denote $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ the training set containing N samples and labels drawn from a joint distribution $P_{(X,Y)}$. The input $\mathbf{x}_i \in \mathbb{R}^d$ is processed by a neural network $f_{\boldsymbol{\omega}}$, of parameters $\boldsymbol{\omega}$, that outputs classification predictions $\hat{y}_i = f_{\boldsymbol{\omega}}(\mathbf{x}_i) \in \mathbb{R}$.

From MLE to MAP. In our context, $P(Y = y_i \mid X = \mathbf{x}_i, \boldsymbol{\omega})$ is a categorical distribution over the classes within the range of Y. We omit the random variable notation in the following for clarity. The log-likelihood of this distribution typically corresponds to the cross-entropy loss, which practitioners often minimize with stochastic gradient descent to obtain a maximum likelihood estimate (MLE): $\mathcal{L}_{\text{MLE}}(\boldsymbol{\omega}) = -\sum_{(\mathbf{x}_i, y_i)\in\mathcal{D}} \log P(y_i \mid \mathbf{x}_i, \boldsymbol{\omega})$.

Going further, the Bayesian framework allows us to incorporate prior knowledge regarding $\boldsymbol{\omega}$ denoted as the distribution $P(\boldsymbol{\omega})$ that complements the likelihood and leads to the research of the maximum a posteriori (MAP), via the minimization of the following loss function:

$$\mathcal{L}_{\text{MAP}}(\boldsymbol{\omega}) = -\sum_{(\mathbf{x}_i, y_i)\in\mathcal{D}} \log P(y_i \mid \mathbf{x}_i, \boldsymbol{\omega}) - \log P(\boldsymbol{\omega}). \quad (1)$$

The normal prior is the standard choice for $P(\boldsymbol{\omega})$, leading to the omnipresent L2 weight regularization.

3.2. Bayesian Posterior and BNNs

Typically, DNNs retain a single set of weights $\boldsymbol{\omega}_{\text{MAP}}$ at the end of the training to use at inference. As such, we de facto consider this model as an oracle. In contrast, BNNs attempt to model the posterior distribution $P(\boldsymbol{\omega} \mid \mathcal{D})$ to take all possible models into account. The prediction y for a new sample $\mathbf{x}$ is computed as the expected outcome from an infinite ensemble, including all possible weights sampled from the posterior distribution:

$$P(y \mid \mathbf{x}, \mathcal{D}) = \int_{\boldsymbol{\omega}\in\Omega} P(y \mid \mathbf{x}, \boldsymbol{\omega}) P(\boldsymbol{\omega} \mid \mathcal{D}) d\boldsymbol{\omega}. \quad (2)$$

However, in practice, this Bayes ensemble approach is intractable since the integral Eq. (2) is computed over the entire parameter space Ω. Practitioners [51] approximate this integral by averaging predictions derived from a finite set $\{\boldsymbol{\omega}_1, \dots \boldsymbol{\omega}_M\}$ of M weight configurations sampled from the posterior distribution:

$$P(y \mid \mathbf{x}, \mathcal{D}) \approx \frac{1}{M} \sum_{m=1}^{M} P(y \mid \mathbf{x}, \boldsymbol{\omega}_m). \quad (3)$$

Let us start with a simple Multi-Layer Perceptron (MLP) with two hidden layers, without loss of generality. For a given input data point $\mathbf{x}$, the prediction of the DNN is defined as follows:

$$\begin{aligned}
\mathbf{h}_1 &= W^{(1)}\mathbf{x} \\
\mathbf{u}_1 &= \text{norm}(\mathbf{h}_1, \beta_1, \gamma_1) = \frac{\mathbf{h}_1 - \hat{\mu}_1}{\hat{\sigma}_1} \times \gamma_1 + \beta_1 \\
\mathbf{a}_1 &= a(\mathbf{u}_1) \\
\mathbf{u}_2 &= \text{norm}\left(W^{(2)}\mathbf{a}_1, \beta_2, \gamma_2\right), \text{ and } \mathbf{a}_2 = a(\mathbf{u}_2) \\
\mathbf{h}_3 &= W^{(3)}\mathbf{a}_2, \text{ and } P(y \mid \mathbf{x}, \boldsymbol{\omega}) = \text{soft}(\mathbf{h}_3),
\end{aligned} \quad (4)$$

where $\mathbf{h}_1$ and $\mathbf{h}_2$ are the preactivation maps and $a(\cdot)$ the activation function. In Eq. (4), soft$(\cdot)$ is the softmax, $\mathbf{a}_1$ and $\mathbf{a}_2$ are the hidden activations, and $\{W^{(j)}\}_{j\in\{0,1,2\}}$ correspond to the weights of the linear layers. The operator norm$(\cdot, \beta_j, \gamma_j)$, of trainable parameters β_j and γ_j, can refer to any batch, layer, or instance normalization (BN, LN, IN). Finally, norm comes with its empirical mean $\hat{\mu}_{\mathbf{u}_j}$ and variance $\hat{\sigma}_{\mathbf{u}_j}$. We omit the small value often added for computational stability.

In the current form, we can leverage this architecture to learn different tasks. However, modeling the uncertainty of the predictions beyond the use of softmax scores as confidence proxies is non-trivial. BNNs [5] are one solution to improve uncertainty estimation. Generally, they hypothesize the independence of the layers and sample from the resulting posterior estimate. For the j-th layer, this yields:

$$\begin{aligned} &\mathbf{u}_j = \mathrm{norm}(W^{(j)}\mathbf{x}, \beta_j, \gamma_j), W^{(j)} \sim P(W^{(j)}|\mathcal{D}), \\ &\mathbf{a}_j = a(\mathbf{u}_j). \end{aligned} \tag{5}$$

As such, BNNs approximate the marginalization (2) of the parameters – an extremely complex task – by generating multiple predictions. Variational inference BNNs [5], the most scalable version among these methods, base their estimation on the "reparametrization trick", here at layer j:

$$\begin{aligned} &\mathbf{u}_j = \mathrm{norm}\left(\left[W_\mu^{(j)} + \boldsymbol{\epsilon}_j W_\sigma^{(j)}\right]\mathbf{h}_{j-1}, \beta_j, \gamma_j\right), \text{ and} \\ &\mathbf{a}_j = a(\mathbf{u}_j), \end{aligned} \tag{6}$$

where the matrices $W_\mu^{(j)}$ and $W_\sigma^{(j)}$ denote the mean and standard deviation of the posterior distribution of layer j, and $\boldsymbol{\epsilon}_j \sim \mathcal{N}(\mathbf{0}, \mathbb{1})$ is a zero-mean unit-diagonal Gaussian vector or matrix. This method enables learning an estimate of a diagonal posterior distribution at the cost of tripling the number of parameters compared to a standard network.

4. ABNN

4.1. Converting DNNs into BNNs

We base our post-hoc Bayesian strategy on pre-trained DNNs that incorporate normalization layers such as batch [43], layer [3], or instance normalization [84]. This is not a limiting factor as most modern architectures include one type of these layers [18, 31, 57]. Subsequently, we modify these normalization layers by introducing a Gaussian perturbation, incorporating our novel Bayesian Normalization Layer (BNL). This adaptation aims to transform the initially deterministic DNN into a BNN. The introduction of the BNL allows us to efficiently leverage pre-trained models, facilitating the conversion to a BNN with minimal alterations. We propose replacing the normalization layers with our novel Bayesian normalization layers (BNL) that incorporate Gaussian noise to transform the deterministic DNNs into BNNs easily. BNLs unlock the power of pre-trained models for uncertainty-aware Bayesian networks. Formally, our BNN is defined as:

$$\begin{aligned} &\mathbf{u}_j = \mathbf{BNL}\left(W^{(j)}\mathbf{h}_{j-1}\right), \text{ and} \\ &\mathbf{a}_j = a(\mathbf{u}_j) \text{ with} \\ &\mathbf{BNL}(\mathbf{h}_j) = \frac{\mathbf{h}_j - \hat{\mu}_j}{\hat{\sigma}_j} \times \gamma_j(1 + \boldsymbol{\epsilon}_j) + \beta_j. \end{aligned} \tag{7}$$

The empirical mean and variance are still represented by $\hat{\mu}_{\mathbf{u}_j}$ and $\hat{\sigma}_{\mathbf{u}_j}$ and computed through batch, layer or instance normalization. In the equation, $\boldsymbol{\epsilon}_j \sim \mathcal{N}(\mathbf{0}, \mathbb{1})$ signifies a sample drawn from a Normal distribution, and γ_j and β_j are the two learnable vectors of the j-th layer.

The DNN being transformed into a BNN, we exclusively retrain the parameters γ_j and β_j for a limited number of epochs using the loss introduced in Section 4.2. To further improve its reliability and generalization properties [90], we do not train a singular ABNN, but rather multiple copies of ABNNs, as explained in section 4.2, resulting in a finite set $\boldsymbol{\omega}_1, \ldots \boldsymbol{\omega}_M$ of M weight configurations. We discuss the benefits of this multi-modality in Appendix A.2.

During inference, for each sample from ABNN $\boldsymbol{\omega}_m$, we augment the number of samples by independently sampling multiple $\boldsymbol{\epsilon}_j \sim \mathcal{N}(\mathbf{0}, \mathbb{1})$. With $\boldsymbol{\epsilon}$ the concatenation of all $\boldsymbol{\epsilon}_j$, and $\{\boldsymbol{\epsilon}_l\}_{l\in[1,L]}$ the set of $\boldsymbol{\epsilon}$s, each individual ABNN sample is expressed as $P(y \mid x, \omega, \boldsymbol{\epsilon})$. During inference, the prediction y for a new sample $\mathbf{x}$ is computed as the expected outcome from a finite ensemble, encompassing all the weights sampled from the posterior distribution:

$$P(y \mid \mathbf{x}, \mathcal{D}) \approx \frac{1}{ML}\sum_{l=1}^{L}\sum_{m=1}^{M} P(y \mid \mathbf{x}, \boldsymbol{\omega}_m, \boldsymbol{\epsilon}_l). \tag{8}$$

We study the effects of L and M in Section H.

4.2. ABNN training loss

The multimodality [22, 44, 54] of the posterior distribution of DNNs is a challenge to any attempt to perform variational inference with on a mono-modal distribution. Wilson and Izmailov [90] have proposed to tackle this issue by training multiple BNNs. However, such approaches inherit the instability of classical BNNs and may struggle to capture different modes accurately. ABNN encounters a similar challenge, requiring safeguards against collapsing into the same local minima during post-training. To mitigate this problem, we introduce a small perturbation to the loss function, preventing collapse and encouraging diversity into the training process. This perturbation involves a modification of the class weights within the cross-entropy loss, now defined as:

$$\mathcal{E}(\boldsymbol{\omega}) = -\sum_{(\mathbf{x}_i, y_i)\in\mathcal{D}} \eta_i \log P(y_i \mid \mathbf{x}_i, \boldsymbol{\omega}). \tag{9}$$

In this formula, η_i represents the class-dependent random weight we initialize at the beginning of training. Typically,

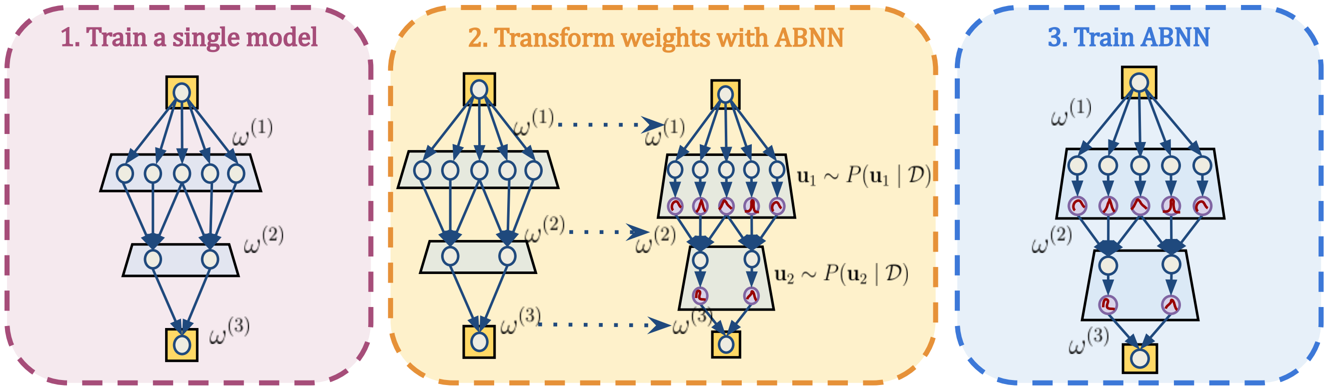

1. Train a single model
$\omega^{(1)}$
$\omega^{(2)}$
$\omega^{(3)}$
2. Transform weights with ABNN
$\mathbf{u}_1 \sim P(\mathbf{u}_1 \mid \mathcal{D})$
$\mathbf{u}_2 \sim P(\mathbf{u}_2 \mid \mathcal{D})$
3. Train ABNN

Figure 2. **Illustration of the training process for the ABNN.** The procedure begins with training a single DNN $\boldsymbol{\omega}_{\text{MAP}}$, followed by architectural adjustments to transform it into an ABNN. The final step involves fine-tuning the ABNN model.

it can be set to zero or one to amplify the effect of certain classes. In contrast to classical variational BNNs [5] that optimize the evidence lower-bound loss, ABNN maximizes the MAP. The optimization involves the following loss:

$$\mathcal{L}(\boldsymbol{\omega}) = \mathcal{L}_{\text{MAP}}(\boldsymbol{\omega}) + \mathcal{E}(\boldsymbol{\omega}). \tag{10}$$

Drawing inspiration from Fort et al. [22], we introduce class-dependent random weight to enhance ensemble diversity. For ABNN ensembles, we introduce a bias through $\mathcal{E}$, aimed at augmenting diversity while minimally impacting BNN predictions. Specifically, we randomly assign a portion of classes to be intentionally prioritized (*i.e.,* akin to creating a form of experts), and amplify the loss associated with these classes by a factor of 5 or 7.

4.3. ABNN training procedure

ABNN is trained through a post-hoc process designed to leverage the strength of Bayesian concepts, improving the uncertainty prediction of DNNs. The training pseudo code for ABNN – in Alg. 1 – details the transformation of conventional DNNs into ensembles of Bayesian models.

We start from a pre-trained neural network (Alg. 1, line 1) and introduce the Bayesian normalization layers, replacing the old batch, instance, or layer normalization layers of the former DNN (Alg. 1 lines 2 to 10). This operation transforms the conventional deterministic network into a BNN to help quantify uncertainty. We initialize the weights of the new layers with the values of the original layers.

Then, we fine-tune the modified network to capture better the inherent uncertainty (Alg. 1 lines 12 to 24). The full process is described in Figure 4, providing a clear overview of the modifications made to enable Bayesian modeling. To improve the posterior estimation of our ABNN models, we fine-tune multiple instances of the normalization layers (typically 3 to 4). This ensemble approach provides robustness and contributes to a more reliable estimation of the posterior distribution. Training multiple ABNNs, each starting from the same checkpoint, enhances our ability to capture diverse modes of the true posterior, thereby improving the overall uncertainty quantification.

Algorithm 1 ABNN training procedure

```
1:  $f_{\boldsymbol{\omega}_{\text{MAP}}}$, : pre-trained network ,$\lambda$ : learning rate,
    nb_epoch : number of epoch
2:  (Step 2: adapt the DNN to a DNN)
3:  # Build a list of all the normalisation layers
4:  normalisation=[Batch_normalisation, Layer_normalisation,
    Instance_normalisation]
5:  for $layer \in$ f$_{\boldsymbol{\omega}_{\text{MAP}}}$.layers : to do
6:      begin
7:      (Transform all Normalization Layers)
8:      if (layer $\in normalisation$) : then
9:          replace layer by BNL
10:     end
11: (Step 3: train ABNN)
12: t=0
13: for $(epoch) \in$ nb_epoch : to do
14:     begin
15:     for $(x, y) \in$ trainloader : to do
16:         begin
17:         (Forward pass)
18:         $\forall x_i \in B(t)$ calculate $f_{\boldsymbol{\omega}(t)}(x_i)$
19:         evaluate the loss $\mathcal{L}(\boldsymbol{\omega}(t), B(t))$
20:         $\boldsymbol{\omega}(t) \leftarrow \boldsymbol{\omega}(t-1) - \lambda \nabla \mathcal{L}_{\boldsymbol{\omega}(t)}$
21:         (Step update)
22:         $t \leftarrow t+1$
23:         end
24:     end
```

4.4. Theoretical analysis

Our approach raises several theoretical questions. In the supplementary material – as detailed in Section A.1 – we show that ABNN exhibits greater stability than classical BNNs. Indeed, in variational inference BNNs, the gradients vary greatly: $\boldsymbol{\epsilon}$, crucial for the Bayesian interpretation,

		CIFAR-10						CIFAR-100						
	Method	**Acc** ↑	**NLL** ↓	**ECE** ↓	**AUPR** ↑	**AUC** ↑	**FPR95** ↓	**Acc** ↑	**NLL** ↓	**ECE** ↓	**AUPR** ↑	**AUC** ↑	**FPR95** ↓	**Time** (h) ↓
ResNet-50	Single Model	95.1	0.211	3.1	95.2	91.9	23.6	78.3	0.905	8.9	87.4	77.9	57.6	**1.7**
	BatchEnsemble	93.9	0.255	3.3	94.7	91.3	20.1	66.6	1.788	18.2	85.2	74.6	60.6	17.2
	MIMO ($\rho = 1$)	95.4	0.197	3.0	95.1	90.8	26.0	79.0	0.876	7.9	87.5	76.9	64.7	6.7
	LPBNN	94.3	0.231	2.3	92.7	86.7	54.9	78.5	1.02	11.3	88.2	77.8	73.5	17.2
	MCDropout	94.4	0.190	1.9	93.1	86.9	43.8	76.9	0.858	3.9	87.8	77.1	64.1	**1.7**
	MCBN	95.0	0.168	**0.7**	95.7	92.6	20.1	78.4	0.83	2.8	86.8	77.5	57.7	**1.7**
	Deep Ensembles	**96.0**	**0.136**	0.8	**97.0**	**94.7**	**15.5**	**80.9**	**0.713**	**2.6**	89.2	80.8	52.5	6.8
	Laplace	95.3	0.160	1.3	96.0	93.3	18.8	78.2	0.99	14.2	89.2	81.0	51.8	**1.7**
	ABNN	95.0	0.160	1.0	96.5	93.9	17.5	77.8	0.828	4.5	**90.0**	**82.0**	**51.3**	2.0
WideResNet-28×10	Single Model	95.4	0.200	2.9	96.1	93.2	20.4	80.3	0.963	15.6	81.0	64.2	80.1	**4.2**
	BatchEnsemble	95.6	0.206	2.7	95.5	92.5	22.1	82.3	0.835	13.0	88.1	78.2	69.8	25.6
	MIMO ($\rho = 1$)	94.7	0.234	3.4	94.9	90.6	30.9	80.2	0.822	2.8	84.9	72.0	72.8	12.6
	LPBNN	95.1	0.249	2.9	95.4	91.2	29.5	79.7	0.831	7.0	79.0	70.1	71.4	23.3
	MCDropout	95.7	0.138	0.6	96.2	93.5	12.8	79.2	0.758	**1.2**	**89.4**	**80.1**	58.6	**4.2**
	MCBN	95.5	**0.133**	**0.5**	96.5	94.2	14.6	80.4	0.749	1.4	80.4	67.8	63.1	**4.2**
	Deep Ensembles	**95.8**	0.143	1.3	**97.8**	**96.0**	**12.5**	**82.5**	0.903	22.9	81.6	67.9	71.3	16.6
	Laplace	95.6	0.151	0.8	95.0	90.7	31.9	80.1	0.942	16.0	83.4	72.1	59.9	**4.2**
	ABNN	95.3	0.146	1.1	96.9	94.6	16.3	80.4	**0.734**	3.4	88.9	78.7	**58.0**	5.0

Table 1. **Performance comparison on CIFAR-10/100 using ResNet-50 and WideResNet28×10** (averaged over multiple runs). All ensembles have $M = 3$ subnetworks, and $L = 3$ for ABNN. We highlight the best performances in bold. Time is the training time in hours on a single RTX 3090, similar for CIFAR-10 and CIFAR-100.

introduces instabilities, perturbating the training. ABNN reduces this burden by applying this term on the latent space rather than the weights, thereby reducing the variance of the gradients, as empirically demonstrated in Appendix A.1.

Another question concerns the theoretical need to modify the loss and add the second term $\mathcal{E}$. We show in Appendix A.2 that it is theoretically sound in the case of a convex problem. Given that DNN optimization is inherently non-convex, adding this term may be theoretically debatable. However, a sensitivity analysis of this term – developed in Appendix D – shows empirical benefits for performance and uncertainty quantification

Finally, we discuss the challenge of estimating the equivalent BNNs to our networks in Appendix A.3. Despite the theoretical value this information could provide concerning the posterior, it remains unused in practice. We solely sample the ϵ and average over multiple training terms to generate robust predictions during inference.

5. Experiments & Results

We test ABNN on image classification and semantic segmentation tasks. For each task, dataset, and architecture, we report metrics relative to the performance of the models but also measure their uncertainty quantification abilities. All our models are implemented in PyTorch and lightning and trained on a single Nvidia RTX 3090 using the TorchUncertainty framework. Appendix G details the hyper-parameters used in our experiments across the different architectures and datasets.

5.1. Image classification

Datasets. We demonstrate the efficiency of ABNN on different datasets and backbones. We start with CIFAR-10 and CIFAR-100 [49] with ResNet-50 [31] and WideResNet28-10 [95]. We then report results for ABNN on ImageNet [15] with ResNet-50 and ViT [18]. In the former case, we train all models from scratch. In the latter, we start from torchvision pre-trained models [68].

Baselines. We compare ABNN against Deep Ensembles [51] and four other ensembles: BatchEnsemble [87], MIMO [30], Masksembles [19], and Laplace [12]. Additionally, we include MCBN [80], and MCDropout [26].

Metrics. We evaluate the performance on classification tasks with the accuracy (Acc) and the Negative Log-Likelihood (NLL). We complete these metrics with the expected top-label calibration error (ECE) [61] and measure the quality of the OOD detection using the Areas Under the Precision/Recall curve (AUPR) and the operating Curve (AUC), as well as the False Positive Rate at 95% recall (FPR95) similarly to Hendrycks et al. [34]. We express all metrics in %.

OOD detection datasets. For OOD detection tasks on CIFAR-10 and CIFAR-100, we use the SVHN dataset [65] as the out-of-distribution dataset and transform the initial problem into binary classification between in-distribution and out-of-distribution data using the maximum softmax probability as the criterion. For ImageNet, we use Describable Texture [85] as the out-of-distribution dataset.

Results. Tables 1 and 2 present the performance of ABNN across various architectures for CIFAR-10/100 and ImageNet, respectively. Notably, ABNN consistently surpasses

	Method	Acc ↑	ECE ↓	AUPR ↑	AUC ↑	FPR95 ↓
ResNet-50	Single Model	77.8	12.1	18.0	80.9	68.6
	BatchEnsemble	75.9	**3.5**	**20.2**	81.6	66.5
	MIMO ($\rho = 1$)	77.6	14.7	18.4	81.6	66.8
	Deep Ensembles	79.2	23.3	19.6	**83.4**	**62.1**
	Laplace	**80.4**	44.3	13.9	75.9	82.8
	ABNN	79.5	9.65	17.8	82.0	65.2
ViT	Single Model	80.0	5.2	19.5	84.1	58.5
	Deep Ensembles	**81.7**	13.5	21.7	**85.5**	60.3
	Laplace	81.0	10.8	**22.1**	83.1	70.6
	ABNN	80.6	**4.32**	21.7	85.4	**55.1**

Table 2. **Performance on ImageNet using ResNet-50 and ViT** concerning in distribution and out-of-distribution metrics.

Laplace approximation and single-model baselines on most datasets and architectures. Furthermore, ABNN exhibits competitive performance compared to Deep Ensembles, achieving equivalent results with a similar training time to a single model. These findings underscore ABNN as a powerful and efficient method, demonstrating superior uncertainty quantification capabilities in image classification tasks while being easier to train.

5.2. Semantic segmentation

For the semantic segmentation part, we compare ABNN against MCP [34], Deep Ensembles [51], MC Dropout [26], TRADI [23], MIMO [30] and LP-BNN [25], on StreetHazards [35], BDD-Anomaly [35], and MUAD [24] that allows comparison on diverse uncertainty quantification aspects of semantic segmentation.

StreetHazards [35]. StreetHazards is a large-scale synthetic dataset comprising various images depicting street scenes. The dataset consists of $5,125$ images for training and an additional $1,500$ images for testing. The training dataset has pixel-wise annotations available for 13 different classes. The test dataset is designed with 13 classes seen during training and an additional 250 out-of-distribution (OOD) classes that were not part of the training set. This diverse composition allows for assessing the algorithm's robustness in the face of various potential scenarios. In our experiments, we employed DeepLabv3+ with a ResNet-50 encoder, as introduced by Chen et al. [8].

BDD-Anomaly [35]. BDD-Anomaly, a subset of the BDD100K dataset [92], comprises $6,688$ street scenes for training and an additional 361 for the test set. Within the training set, pixel-level annotations are available for 17 distinct classes. The test dataset consists of the same 17 classes seen during training and introduces 2 out-of-distribution (OOD) classes: motorcycle and train. In our experimental setup, we adopted DeepLabv3+[8] and followed the experimental protocol outlined in[35]. Similar to previous experiments, we utilized a ResNet-50 encoder [31] for the neural network architecture.

	Method	mIoU ↑	AUPR ↑	AUC ↑	FPR95 ↓	ECE ↓
StreetHazards	Single Model	53.90	6.91	86.60	35.74	6.52
	TRADI	52.46	6.93	87.39	38.26	6.33
	Deep Ensembles	55.59	**8.32**	87.94	**30.29**	5.33
	MIMO	55.44	6.90	87.38	32.66	5.57
	BatchEnsemble	**56.16**	7.59	88.17	32.85	6.09
	LP-BNN	54.50	7.18	88.33	32.61	**5.20**
	ABNN	53.82	7.85	**88.39**	32.02	6.09
BDD-Anomaly	Single Model	47.63	4.50	85.15	28.78	17.68
	TRADI	44.26	4.54	84.80	36.87	16.61
	Deep Ensembles	**51.07**	5.24	84.80	**28.55**	14.19
	MIMO	47.20	4.32	84.38	35.24	16.33
	BatchEnsemble	48.09	4.49	84.27	30.17	16.90
	LP-BNN	49.01	4.52	85.32	29.47	17.16
	ABNN	48.76	**5.98**	**85.74**	29.01	**14.03**
MUAD	Single Model	57.32	26.04	86.24	39.43	6.07
	MC-Dropout	55.62	22.25	84.39	45.75	6.45
	Deep Ensembles	58.29	**28.02**	87.10	37.60	5.88
	BatchEnsemble	57.10	25.70	86.90	38.81	6.01
	MIMO	57.10	24.18	86.62	34.80	5.81
	ABNN	**61.96**	24.37	**91.55**	**21.68**	**5.58**

Table 3. **Comparative results on the OOD task for semantic segmentation.** We run all methods in similar settings using publicly available code for related methods and average over three seeds. The architecture is a DeepLabv3+ based on ResNet50.

MUAD [24]. MUAD consists of 3,420 images in the training set and 492 in the validation set. The test set comprises 6,501 images, distributed across various subsets: 551 in the normal set, 102 in the normal set with no shadow, and 1,668 in the out-of-distribution (OOD) set. All these sets cover both day and night conditions, with a distribution of 2/3 day images and 1/3 night images. MUAD encompasses 21 classes, with the initial 19 classes mirroring those found in CityScapes [10]. Additionally, three classes are introduced to represent object anomalies and animals, adding diversity to the dataset. In our first experiment, we employed a DeepLabV3+ [8] network with a ResNet50 encoder[31] for training on MUAD.

Results. Table 3 presents the results of ABNN, compared to various baselines on the three datasets. ABNN performs competitively with Deep Ensembles, a technique known for accurately quantifying uncertainty. Moreover, our approach exhibits faster training times, making it potentially more appealing for practitioners. We have not included a comparison with Laplace approximation [12], as it is not commonly applied to semantic segmentation, and adapting DNNs for Laplace Approximation is not straightforward.

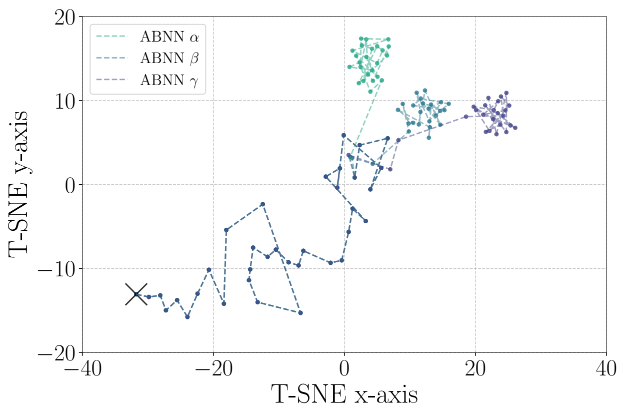
6. Discussions

6.1. General discussions

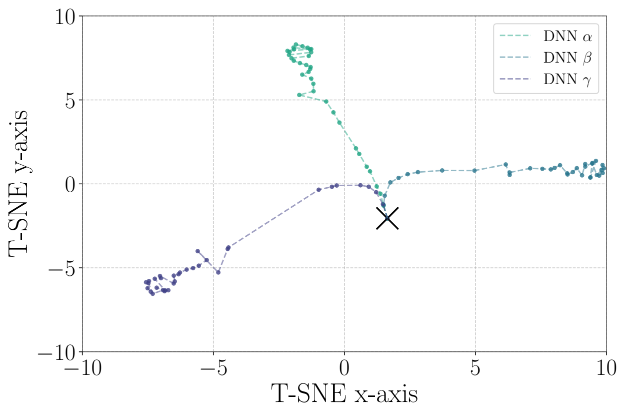
We develop several discussions in the supplementary materials. First, we explore the theoretical aspects, including the stability of the DNNs in Appendix A.1, the importance of multi-mode in Section A.2, and the relationship with classical BNNs in Appendix A.3. We experiment on the transfer of ViT-B-16 from Imagenet 21k [71] to CIFAR-100 which highlights the potential of ABNN in transfer learning, achieving an accuracy of 92.18%. Additionally, we perform several ablation studies, notably on the impact of discarding multi-mode or the loss term $\mathcal{E}$ (defined in Section 4.2) in Appendix D. We show that discarding $\mathcal{E}$ reduces the performance while incorporating the multi-mode improves uncertainty quantification. Moreover, in Section B, we analyze the variance of the gradients, confirming that our technique exhibits lower gradients than BNNs, making it more stable and easier to train. Finally, Appendix C delves into the variability of our method under different scenarios, exploring cases where we initiate ABNN from a single model and optimize from various initial checkpoints. Although our technique inherits the instabilities of the DNN, we observe that the standard variation is five times larger than that of the single model, indicating less stability than a standard DNN.

6.2. Diversity of ABNN

Concerning diversity, we train ABNN on CIFAR-10 using a ResNet-50 architecture Specifically, we optimize a ResNet for 200 epochs and then fine-tune three ABNNs, starting from the optimal checkpoint. Additionally, we train two other ResNet-50s on CIFAR-10 to form proper Deep Ensembles. As depicted in Figure 3, ABNN does not exhibit the same level of diversity as the Deep Ensembles. However, it is intriguing that even when initiated from a single DNN, ABNN manages to depart from its local minimum and explore different modes. This concept of different modes is further supported by Section E, where we analyze the mutual information of various ABNN checkpoints.



(a) ABNN



(b) Deep Ensembles

Figure 3. **Comparison of the diversities of ABNN and Deep Ensembles [51].** T-SNE plot of the 20 principal components of the logits from 384 images for ABNN (a) and Deep Ensembles (b).

7. Conclusion

Our approach, ABNN, introduces a novel perspective to uncertainty quantification. Leveraging the strengths of pre-trained deterministic models, ABNN strategically transforms them into Bayesian networks with minimal modifications, offering enhanced stability and efficient posterior exploration. Through comprehensive experimental analyses, we demonstrate the effectiveness of ABNN both in predictive performance and uncertainty quantification, showcasing its potential applications in diverse scenarios.

The multi-mode characteristic of ABNN, coupled with a carefully designed loss function, not only addresses the challenge of the multi-modality of the posterior but also provides a stable and diverse ensemble of models. Our empirical evaluations on various datasets and architectures highlight the superiority of ABNN over traditional BNNs and post-hoc uncertainty quantification methods such as the Laplace approximation and showcase its competitiveness compared to state-of-the-art such as Deep Ensembles.

Moreover, ABNN exhibits promising results in transfer learning scenarios, underscoring its potential for broader applications. The insights gained from theoretical discussions and ablation studies further elucidate the underlying mechanisms of ABNN, contributing to a deeper understanding of its behavior and performance.

In summary, ABNN emerges as a robust and flexible solution for uncertainty-aware deep learning, offering a pragmatic bridge between deterministic and Bayesian paradigms. Its simplicity in implementation, coupled with superior performance and stability, positions ABNN as a valuable tool in the contemporary landscape of machine learning and Bayesian modeling.

Acknowledgments. This work was partially supported by ELSA - European Lighthouse on Secure and Safe AI funded by the European Union under grant agreement No. 101070617. This work was performed using HPC resources from GENCI-IDRIS (Grant 2023-[AD011011970R3]).

References

[1] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, et al. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *IF*, 2020. 1

[2] Arsenii Ashukha, Alexander Lyzhov, Dmitry Molchanov, and Dmitry Vetrov. Pitfalls of in-domain uncertainty estimation and ensembling in deep learning. In *ICLR*, 2019. 15

[3] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. In *NeurIPSW*, 2016. 4

[4] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *JASA*, 2017. 3

[5] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural network. In *ICML*, 2015. 2, 3, 4, 5, 12

[6] Nicolas Brosse, Carlos Riquelme, Alice Martin, Sylvain Gelly, and Éric Moulines. On last-layer algorithms for classification: Decoupling representation from uncertainty estimation. *arXiv preprint arXiv:2001.08049*, 2020. 2

[7] Robin Chan, Krzysztof Lis, Svenja Uhlemeyer, Hermann Blum, Sina Honari, Roland Siegwart, Pascal Fua, Mathieu Salzmann, and Matthias Rottmann. Segmentmeifyoucan: A benchmark for anomaly segmentation. In *NeurIPS*, 2021. 1

[8] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, 2018. 7

[9] Charles Corbière, Marc Lafon, Nicolas Thome, Matthieu Cord, and Patrick Pérez. Beyond first-order uncertainty estimation with evidential models for open-world recognition. In *ICMLW*, 2021. 2

[10] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016. 7

[11] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *CVPR*, 2020. 16

[12] Erik Daxberger, Agustinus Kristiadi, Alexander Immer, Runa Eschenhagen, Matthias Bauer, and Philipp Hennig. Laplace redux–effortless Bayesian deep learning. In *NeurIPS*, 2021. 2, 6, 7

[13] Erik Daxberger, Eric Nalisnick, James U Allingham, Javier Antorán, and José Miguel Hernández-Lobato. Bayesian deep learning via subnetwork inference. In *ICML*, 2021. 2

[14] Mostafa Dehghani, Josip Djolonga, Basil Mustafa, Piotr Padlewski, Jonathan Heek, Justin Gilmer, Andreas Peter Steiner, Mathilde Caron, Robert Geirhos, Ibrahim Alabdulmohsin, et al. Scaling vision transformers to 22 billion parameters. In *ICML*, 2023. 3

[15] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 2, 6

[16] Stefan Depeweg, Jose-Miguel Hernandez-Lobato, Finale Doshi-Velez, and Steffen Udluft. Decomposition of uncertainty in bayesian deep learning for efficient and risk-sensitive learning. In *ICML*, 2018. 2

[17] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 1

[18] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021. 1, 4, 6

[19] Nikita Durasov, Timur Bagautdinov, Pierre Baque, and Pascal Fua. Masksembles for uncertainty estimation. In *CVPR*, 2021. 6

[20] 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, 12, 14

[21] William Feller. *An introduction to probability theory and its applications, Volume 2*. John Wiley & Sons, 1991. 14

[22] Stanislav Fort, Huiyi Hu, and Balaji Lakshminarayanan. Deep ensembles: A loss landscape perspective. *arXiv preprint arXiv:1912.02757*, 2019. 2, 4, 5, 15

[23] Gianni Franchi, Andrei Bursuc, Emanuel Aldea, Séverine Dubuisson, and Isabelle Bloch. TRADI: Tracking deep neural network weight distributions. In *ECCV*, 2020. 2, 3, 7

[24] Gianni Franchi, Xuanlong Yu, Andrei Bursuc, Angel Tena, Rémi Kazmierczak, Séverine Dubuisson, Emanuel Aldea, and David Filliat. MUAD: Multiple uncertainties for autonomous driving, a benchmark for multiple uncertainty types and tasks. In *BMVC*, 2022. 2, 7

[25] Gianni Franchi, Andrei Bursuc, Emanuel Aldea, Séverine Dubuisson, and Isabelle Bloch. Encoding the latent posterior of bayesian neural networks for uncertainty quantification. *T-PAMI*, 2023. 2, 3, 7

[26] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *ICML*, 2016. 2, 3, 6, 7

[27] Jakob Gawlikowski, Cedrique Rovile Njieutcheu Tassi, Mohsin Ali, Jongseok Lee, Matthias Humt, Jianxiang Feng, Anna Kruspe, Rudolph Triebel, Peter Jung, Ribana Roscher, et al. A survey of uncertainty in deep neural networks. *AI Review*, 2023. 2

[28] Ethan Goan and Clinton Fookes. Bayesian neural networks: An introduction and survey. *Case Studies in Applied Bayesian Data Science*, 2020. 2

[29] Lars Kai Hansen and Peter Salamon. Neural network ensembles. *T-PAMI*, 1990. 3

[30] Marton Havasi, Rodolphe Jenatton, Stanislav Fort, Jeremiah Zhe Liu, Jasper Snoek, Balaji Lakshminarayanan, Andrew Mingbo Dai, and Dustin Tran. Training independent subnetworks for robust prediction. In *ICLR*, 2021. 2, 3, 6, 7

[31] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2, 4, 6, 7

[32] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017. 1

[33] Matthias Hein, Maksym Andriushchenko, and Julian Bitterwolf. Why relu networks yield high-confidence predictions far away from the training data and how to mitigate the problem. In *CVPR*, 2019. 1

[34] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *ICLR*, 2017. 6, 7

[35] Dan Hendrycks, Steven Basart, Mantas Mazeika, Mohammadreza Mostajabi, Jacob Steinhardt, and Dawn Song. A benchmark for anomaly segmentation. *arXiv preprint arXiv:1911.11132*, 2019. 2, 7

[36] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, et al. Jacob steinhardt et justin gilmer. the many faces of robustness: A critical analysis of out-of-distribution generalization. In *ICCV*, 2021. 1

[37] Dan Hendrycks, Nicholas Carlini, John Schulman, and Jacob Steinhardt. Unsolved problems in ml safety. *arXiv preprint arXiv:2109.13916*, 2021. 1

[38] José Miguel Hernández-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *ICML*, 2015. 3

[39] Stephen C Hora. Aleatory and epistemic uncertainty in probability elicitation with an example from hazardous waste management. *Reliability Engineering & System Safety*, 1996. 2

[40] 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. 14

[41] Eyke Hüllermeier and Willem Waegeman. Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *ML*, 2021. 2

[42] Gabriel Ilharco, Mitchell Wortsman, Ross Wightman, Cade Gordon, Nicholas Carlini, Rohan Taori, Achal Dave, Vaishaal Shankar, Hongseok Namkoong, John Miller, Hannaneh Hajishirzi, Ali Farhadi, and Ludwig Schmidt. Openclip, 2021. 3

[43] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015. 4

[44] Pavel Izmailov, Sharad Vikram, Matthew D Hoffman, and Andrew Gordon Gordon Wilson. What are bayesian neural network posteriors really like? In *ICML*, 2021. 2, 4

[45] Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *ML*, 1999. 2

[46] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? *NeurIPS*, 2017. 2

[47] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *ICCV*, 2023. 3

[48] Agustinus Kristiadi, Matthias Hein, and Philipp Hennig. Being bayesian, even just a bit, fixes overconfidence in relu networks. In *ICML*, 2020. 2

[49] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, MIT, 2009. 2, 6

[50] A Krizhevsky, I Sutskever, and G Hinton. Imagenet classification with deep convolutional networks. In *NeurIPS*, 2012. 1

[51] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *NeurIPS*, 2017. 2, 3, 6, 7, 8

[52] John Lambert, Ozan Sener, and Silvio Savarese. Deep learning under privileged information using heteroscedastic dropout. In *CVPR*, pages 8886–8895, 2018. 15

[53] Olivier Laurent, Adrien Lafage, Enzo Tartaglione, Geoffrey Daniel, Jean-Marc Martinez, Andrei Bursuc, and Gianni Franchi. Packed-ensembles for efficient uncertainty estimation. In *ICLR*, 2023. 3

[54] Olivier Laurent, Emanuel Aldea, and Gianni Franchi. A symmetry-aware exploration of bayesian neural network posteriors. In *ICLR*, 2024. 4, 13

[55] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *NeurIPS*, 2018. 2

[56] Kaican Li, Kai Chen, Haoyu Wang, Lanqing Hong, Chaoqiang Ye, Jianhua Han, Yukuai Chen, Wei Zhang, Chunjing Xu, Dit-Yan Yeung, et al. Coda: A real-world road corner case dataset for object detection in autonomous driving. In *ECCV*, 2022. 1

[57] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *CVPR*, 2022. 4

[58] David JC MacKay. A practical bayesian framework for backpropagation networks. *Neural computation*, 1992. 2, 3

[59] Wesley J Maddox, Pavel Izmailov, Timur Garipov, Dmitry P Vetrov, and Andrew Gordon Wilson. A simple baseline for bayesian uncertainty in deep learning. In *NeurIPS*, 2019. 2

[60] Andrey Malinin and Mark Gales. Predictive uncertainty estimation via prior networks. In *NeurIPS*, 2018. 2

[61] Mahdi Pakdaman Naeini, Gregory F. Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In *AAAI*, 2015. 6

[62] Eric Thomas Nalisnick. *On priors for Bayesian neural networks*. University of California, Irvine, 2018. 2

[63] Niv Nayman, Avram Golbert, Asaf Noy, Tan Ping, and Lihi Zelnik-Manor. Diverse imagenet models transfer better. *arXiv preprint arXiv:2204.09134*, 2022. 2

[64] Radford M Neal. *Bayesian learning for neural networks*. 2012. 2

[65] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NeurIPSW*, 2011. 6

[66] Yannic Neuhaus, Maximilian Augustin, Valentyn Boreiko, and Matthias Hein. Spurious features everywhere-large-scale detection of harmful spurious features in imagenet. In *ICCV*, 2023. 1

[67] Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, David Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In *NeurIPS*, 2019. 2

[68] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019. 6

[69] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. Technical report, OpenAI, 2018. 1

[70] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016. 1

[71] Tal Ridnik, Emanuel Ben-Baruch, Asaf Noy, and Lihi Zelnik-Manor. Imagenet-21k pretraining for the masses. In *NeurIPS Datasets and Benchmarks*, 2021. 8

[72] Hippolyt Ritter, Aleksandar Botev, and David Barber. A scalable laplace approximation for neural networks. In *ICLR*, 2018. 2, 3

[73] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022. 3

[74] Subhankar Roy, Martin Trapp, Andrea Pilzer, Juho Kannala, Nicu Sebe, Elisa Ricci, and Arno Solin. Uncertainty-guided source-free domain adaptation. In *ECCV*, 2022. 3

[75] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *NeurIPS*, 2022. 3

[76] Aliaksei Severyn and Alessandro Moschitti. Learning to rank short text pairs with convolutional deep neural networks. In *SIGIR*, 2015. 1

[77] Divya Shanmugam, Davis Blalock, Guha Balakrishnan, and John Guttag. Better aggregation in test-time augmentation. In *ICCV*, 2021. 15

[78] Jongwook Son and Seokho Kang. Efficient improvement of classification accuracy via selective test-time augmentation. *IS*, 2023. 15

[79] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016. 16

[80] Mattias Teye, Hossein Azizpour, and Kevin Smith. Bayesian uncertainty estimation for batch normalized deep networks. In *ICML*, 2018. 6

[81] Luke Tierney and Joseph B Kadane. Accurate approximations for posterior moments and marginal densities. *JASA*, 1986. 2, 3

[82] Tishby, Levin, and Solla. Consistent inference of probabilities in layered networks: predictions and generalizations. In *IJCNN*, 1989. 2

[83] Dustin Tran, Jeremiah Liu, Michael W Dusenberry, Du Phan, Mark Collier, Jie Ren, Kehang Han, Zi Wang, Zelda Mariet, Huiyi Hu, et al. Plex: Towards reliability using pretrained large model extensions. *arXiv preprint arXiv:2207.07411*, 2022. 1

[84] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016. 4

[85] Haoqi Wang, Zhizhong Li, Litong Feng, and Wayne Zhang. ViM: Out-of-distribution with virtual-logit matching. In *CVPR*, 2022. 6

[86] Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *ICML*, 2011. 2

[87] Yeming Wen, Dustin Tran, and Jimmy Ba. BatchEnsemble: an alternative approach to efficient ensemble and lifelong learning. In *ICLR*, 2019. 2, 3, 6

[88] Ross Wightman. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019. 16

[89] Ross Wightman, Hugo Touvron, and Herve Jegou. Resnet strikes back: An improved training procedure in timm. In *NeurIPSW*, 2021. 17

[90] Andrew G Wilson and Pavel Izmailov. Bayesian deep learning and a probabilistic perspective of generalization. *NeurIPS*, 2020. 2, 3, 4, 13

[91] Guoxuan Xia and Christos-Savvas Bouganis. Window-based early-exit cascades for uncertainty estimation: When deep ensembles are more efficient than single models. In *ICCV*, 2023. 3

[92] Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *CVPR*, 2020. 7

[93] 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. 16

[94] Éloi Zablocki, Hédi Ben-Younes, Patrick Pérez, and Matthieu Cord. Explainability of deep vision-based autonomous driving systems: Review and challenges. *IJCV*, 2022. 1

[95] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, 2016. 6

[96] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *ICCV*, 2023. 3

[97] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR*, 2018. 16